ARBEITSBERICHTE
DES INSTITUTS FÜR MATHEMATISCHE MASCHINEN UND DATENVERARBEITUNG (INFORMATIK)
FRIEDRICH ALEXANDER UNIVERSITÄT ERLANGEN NÜRNBERG
HERAUSGEBER: H. Billing, A. Bode, W. Händler, U. Herzog, F. Hofmann, K. Leeb, P. Mertens, H. Müller, G. Nees, H. Niemann, B. Schmidt, H. J. Schneider, D. Seitzer, H. Wedekind

ÜBER ORTHOGONALE TRANSFORMATIONEN UND FEHLERKORRIGIERENDE CODES

Walter Fumy

DISSERTATION
BAND 19 · NUMMER 2 · ERLANGEN · JUNI 1986

Diss. Erlangen
1986 № 288

ISSN 0344 - 3515

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41538114_0008

ÜBER ORTHOGONALE TRANSFORMATIONEN UND FEHLERKORRIGIERENDE CODES

Der Technischen Fakultät

der Universität Erlangen-Nürnberg

zur Erlangung des Grades

DOKTOR - INGENIEUR

vorgelegt

von

Walter Fumy

ERLANGEN 1986

Als Dissertation genehmigt von der
Technischen Fakultät der
Universität Erlangen-Nürnberg

Tag der Einreichung: 18.06.1985
Tag der Promotion: 11.04.1986
Dekan: Prof. Dr.-Ing. H. Brand
Berichterstatter: Prof. Dr. K. Leeb
Prof. Dr. Th. Beth
Prof. Dr. H. Müller

für FRANZ STROBEL

1925 - 1984

Vorwort

Orthogonale Transformationen werden heute in nahezu allen Bereichen der digitalen Signalverarbeitung eingesetzt. Zu ihren prominentesten Vertretern zählen zweifellos die Diskrete Fourier-Transformation (DFT) und die Walsh-Hadamard-Transformation (WHT). Stellvertretend für viele weitere Anwendungsgebiete seien Bild- und Sprachverarbeitung, die Merkmalsgewinnung in der Mustererkennung, Analyse und Design von Kommunikationssystemen und die Spektroskopie genannt.

Im Bereich der Codierungstheorie spielten orthogonale Transformationen bis vor kurzem eine eher unauffällige Rolle. Dies belegt z.B. das vor allem zur Bestimmung der Gewichtsverteilung von Codes verwendete 'Mattson-Solomon-Polynom'; hinter dieser Bezeichnung verbirgt sich schlicht die DFT über endlichen Körpern.

In der vorliegenden Arbeit werden verschiedene neue Anwendungsmöglichkeiten orthogonaler Transformationen auf dem Gebiet der fehlerkorrigierenden Codes untersucht. Ein Schwerpunkt liegt dabei auf dem Einsatz von Transformationen für die Implementierung von Codier- und Decodierverfahren. Anstoß für diese Untersuchungen waren die im Rahmen des DFG-Projekts "Algebraische Diskrete Fourier-Transformation" entwickelten Verfahren zur effizienten Berechnung der DFT speziell für den Fall, daß Signal- und Spektralbereich nicht übereinstimmen, sondern der Signalbereich ein Unterkörper des Spektralbereichs ist. Diese Situation ist typisch für viele Anwendungen, insbesondere auch in der Codierungstheorie.

Algebraische Decodierverfahren für über einem endlichen Körper GF(q) definierte Codes (wie etwa der Berlekamp-Algorithmus zur Decodierung von BCH-Codes) benötigen in der Regel Arithmetik in einem Erweiterungskörper $GF(q^m)$ über GF(q). Die bekannten Vorschläge zur Hardware-Implementierung derartiger Arithmetik gehen nahezu ausnahmslos von der Darstellung von $GF(q^m)$ bzgl. einer polynomialen Basis über GF(q) aus. Dagegen beruhen die in Kapitel I vorgestellten Implementierungen verschiedener Verfahren zur Multiplikation und zur Berechnung der multiplikativen Inversen zumeist nicht auf der genannten Standard-Darstellung. Speziell die parallelen Versionen der auf einer Normalbasis-Darstellung basierenden arithmetischen Operationen bieten sich aufgrund ihrer regulären Struktur für VLSI-Implementierungen an.

Die bekannteste Anwendung der Walsh-Hadamard-Transformation in der Codierungstheorie ist das in der 'Green-Machine' realisierte Decodierverfahren für Reed-Muller-Codes, das z.B. von 1969 bis 1977 bei den Mariner-Projekten der NASA eingesetzt wurde. Wie in Kapitel II gezeigt wird, kann die Walsh-Hadamard-Transformation

auch bei der Bestimmung der Gewichtsverteilung linearer Codes und bei der Konstruktion verschiedener Klassen "guter" Codes von Nutzen sein.

Die Fourier-Transformation über endlichen Körpern hat sich in den letzten Jahren als hilfreiches Instrument für die Beschreibung und die Analyse zyklischer Codes erwiesen. Ihre Bedeutung für die Implementierung von Codes ist dagegen gering geblieben. Neuere Arbeiten zeigen Möglichkeiten auf, die DFT aus den Standardverfahren zur Decodierung von BCH-Codes zu eliminieren. Ursache für diese Entwicklung ist der relativ hohe Aufwand zur Berechnung der DFT, obwohl hierfür die bekannten "schnellen" Algorithmen (z.B. die FFT) zur Verfügung stehen. Das in Kapitel III vorgestellte Verfahren zur Berechnung der inversen DFT (und damit zur Codierung zyklischer Codes) stellt einen neuen Zugang zur inversen ADFT dar und zeichnet sich durch günstige Implementierbarkeit aus. Damit ist es geeignet, die Attraktivität DFT-basierter Codier- und Decodierverfahren zu steigern.

Das Konzept der Algebraischen Diskreten Fourier-Transformation (ADFT) führt in Kapitel IV zur Definition einer interessanten neuen Klasse linearer Codes. Diese sogenannten ADFT-Codes umfassen die Klasse der zyklischen Codes im engeren Sinn und besitzen wie diese die für die Implementierung wichtige Eigenschaft, daß die Syndromberechnung im wesentlichen mit derselben Hardware wie die Codierung durchgeführt werden kann.

Im einführenden Kapitel schließlich sind die für diese Arbeit grundlegenden Aspekte der algebraischen Codierungstheorie in knapper Form dargestellt.

An dieser Stelle möchte ich allen danken, die an der Entstehung dieser Arbeit durch Unterstützung in vielen Formen Anteil hatten, sei es, indem sie Anregungen gaben, kritisch zuhörten oder Geduld bewiesen.

Erlangen, im Mai 1986

Notation

In dieser Arbeit werden die folgenden Symbole und Bezeichnungen verwendet:

(x,y)	größter gemeinsamer Teiler von x und y
$[x:y]$	ganzzahliges Intervall zwischen x und y
$[x]$	kleinste ganze Zahl $\geq x$
$\log(x)$	Logarithmus von x zur Basis 2
$\phi(n)$	Euler'sche Funktion
I_n	n-te Einheitsmatrix
S_n	n-te "Spiegelungsmatrix", die Permutation $i \longrightarrow (n-1)\cdot i \bmod n$ vermittelnd
$GF(q)$	endlicher Körper der Ordnung q
$GF(q)^*$	$GF(q) \setminus \{0\}$
$tr(\alpha)$	Spur des Elements α
$deg(f)$	Grad des Polynoms f
$\square$	kennzeichnet das Ende eines Beweises

Algorithmen sind in einer der Sprache 'C' (cf. /Ke78/) entlehnten Notation angegeben.

Inhalt

0.	**Grundlagen**	1
	§1 Lineare Codes	4
	§2 Schranken	8
	§3 Zyklische Codes	12
	§4 Code-Modifikationen	22
	§5 BCH-Codes, Reed-Solomon-Codes und Verallgemeinerungen	25
I.	**Arithmetik in endlichen Körpern**	29
	§1 Basen endlicher Körper	30
	§2 Multiplikationsverfahren	37
	§3 Berechnung von multiplikativen Inversen	53
II.	**Walsh-Hadamard-Transformation und lineare Codes**	59
	§1 Die Walsh-Hadamard-Transformation	60
	§2 WHT und die Gewichtsverteilung linearer Codes	63
	§3 WHT und die Konstruktion linearer Codes	66
III.	**Diskrete Fourier-Transformation und zyklische Codes**	69
	§1 Die Diskrete Fourier-Transformation	70
	§2 DFT und die Beschreibung zyklischer Codes	73
	§3 DFT und die Implementierung zyklischer Codes	79
IV.	**Algebraische DFT und ADFT-Codes**	93
	§1 Die Algebraische Diskrete Fourier-Transformation	94
	§2 Zur Implementierung der ADFT	104
	§3 ADFT-Codes	116
	§4 Zur Implementierung von ADFT-Codes	130
Anhang		139
Literatur		147
Index		158

"a message with content and clarity
has gotten to be quite a rarity
to combat the terror
of serious error
use bits of appropriate parity"

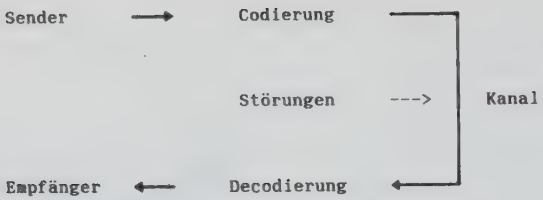
(S.W.Golomb)

0. Grundlagen

Der Ursprung der Codierungstheorie liegt in einem Versprechen der Informationstheorie, dem berühmten Kanalcodierungssatz von C.Shannon /Sh48/:

> Für jede Rate unterhalb der Kanalkapazität und für jedes $\varepsilon > 0$ gibt es ein Codier-/Decodierverfahren derart, daß die Decodierfehlerwahrscheinlichkeit kleiner als ε ist.

Diesem Satz (für den bedauerlicherweise kein konstruktiver Beweis bekannt ist) und den weiteren Betrachtungen liegt das folgende vereinfachte Modell eines Datenübertragungssystems zugrunde:



Kanal steht hierbei für jede Art von räumlicher oder zeitlicher Informationsübertragung. Der Kanal wird als diskret, symmetrisch und gedächtnislos angenommen, seine Symbolfehlerwahrscheinlichkeit und damit seine Kapazität als bekannt vorausgesetzt.

Ein (Block-) ***Code*** der ***Länge*** n ist eine Menge C von n-Tupeln (den ***Codeworten***) über einem Alphabet A.

In diesem Rahmen ist das Alphabet stets ein endlicher Körper GF(q); der Code C ist somit eine Teilmenge des n-dimensionalen Vektorraumes $GF(q)^n$. Die zu codierenden ***Nachrichten*** seien Vektoren fester Länge k (mit $k \leq n$) über GF(q).

Die **Codierung** ist dann eine injektive Abbildung

$$COD: \quad GF(q)^k \dashrightarrow GF(q)^n,$$

die **Decodierung** ist eine surjektive Abbildung

$$DEC: \quad GF(q)^n \dashrightarrow GF(q)^k.$$

Die Schwierigkeit bei der Decodierung besteht im Bestimmen des Fehlermusters, das bei gegebenen Kanaleigenschaften und der Übertragung eines bestimmten Codewortes mit der höchsten Wahrscheinlichkeit aufgetreten ist. Bei einem symmetrischen, gedächtnislosen Kanal ist das wahrscheinlichste Fehlermuster - unabhängig vom übertragenen Codewort - stets eines mit minimaler Anzahl von Koeffizienten ungleich Null.

> Das (Hamming-) **Gewicht** wgt(c) eines Vektors c ist die Anzahl seiner von Null verschiedenen Koeffizienten; der **Hamming-Abstand** dist(u,v) zweier Vektoren u und v ist das Gewicht ihrer Differenz. Die **Minimaldistanz** eines Codes *C* ist der kleinste Hamming-Abstand zweier verschiedener Codeworte.

Die Minimaldistanz d eines Codes *C* bestimmt seine Korrektureigenschaften gegenüber zufällig verteilten Fehlern. *C* ist

- (d-1)-***fehlererkennend*** und
- ((d-1) div 2)-***fehlerkorrigierend***.

Zusätzliche Information über die Positionen möglicher Fehler (die etwa vom Demodulator, von einem Synchronisationsbaustein oder auch von einem vorgeschalteten Code stammen kann) kann vom Decodierer so genutzt werden, daß sich die Fehlerkorrekturfähigkeit eines Codes verbessert (cf. /Fu84/).

§1 Lineare Codes

Ein dem Kanalcodierungssatz von C.Shannon entsprechendes Resultat für den binären symmetrischen Kanal (BSC) und die Klasse der linearen Codes wurde von P.Elias /El54/ bewiesen.

(1.1) **Definition**:

Ein ***linearer Code*** C der Länge n über GF(q) ist ein k-dimensionaler Unterraum von $GF(q)^n$ und wird als ***(n,k)-Code*** bezeichnet. Sind $\underline{a}_0, \underline{a}_1, \ldots, \underline{a}_{r-1}$ Vektoren aus $GF(q)^n$, die C erzeugen (folglich gilt $r \geq k$), so heißt die $(r \times n)$-Matrix $G = (\underline{a}_0, \underline{a}_1, \ldots, \underline{a}_{r-1})^t$ eine ***Generatormatrix*** von C. Der Quotient k/n heißt die ***Rate***, der Quotient $(n-k)/n$ die ***Redundanz*** von C.

(1.2) **Definition**:

Sei C ein (n,k)-Code über GF(q). Der Code

$$C^{\perp} = \{ \underline{u} \in GF(q)^n \mid \underline{u} \cdot \underline{v} = 0 \text{ für alle } \underline{v} \in C \}$$

heißt der ***duale Code*** zu C. $C^{\perp}$ ist ein $(n,n-k)$-Code. Der Code C heißt ***selbst-dual***, falls $C = C^{\perp}$ gilt, ***schwach selbst-dual***, falls $C \subset C^{\perp}$ gilt. Eine Generatormatrix H von $C^{\perp}$ heißt eine ***Kontrollmatrix*** von C.

Die Bezeichnung Kontrollmatrix erklärt sich durch die folgende Eigenschaft:

(1.3) Beobachtung:

Ist H eine Kontrollmatrix eines (n,k)-Codes C über GF(q), so ist ein Vektor $\underline{v} \in GF(q)^n$ genau dann ein Codewort von C, wenn $\underline{v} \cdot H^t = \underline{0}$ gilt.

Zwei (n,k)-Codes über demselben Alphabet GF(q) sind als Vektorräume zwar stets isomorph, können jedoch durchaus verschiedene Minimaldistanz besitzen und sich damit in ihren Fehlerkorrektureigenschaften unterscheiden.

(1.4) Definition:

Zwei (n,k)-Codes C_1 und C_2 über GF(q) heißen **äquivalent**, wenn es eine Permutation π von [0:n-1] gibt, so daß für jedes Codewort $(c_0,c_1,\ldots,c_{n-1})$ des Codes C_1 der Vektor $(c_{\pi(0)},c_{\pi(1)},\ldots,c_{\pi(n-1)}) \in C_2$ ist.

(1.5) Definition:

Sei C ein (n,k)-Code. Eine Codierung, die eine Nachricht $\underline{u} = (u_0,u_1,\ldots,u_{k-1})$ derart auf ein Codewort $\underline{c} = (c_0,c_1,\ldots,c_{n-1})$ abbildet, daß $u_j = c_{i[j]}$ für $0 \leq j < k$ und Indizes $i[0],i[1],\ldots,i[k-1]$ gilt, heißt eine **systematische Codierung**. Gibt es für C eine systematische Codierung für die Indizes $0,1,\ldots,k-1$, so heißt C ein **systematischer Code**.

(1.6) Bemerkungen:

(a) Zu jedem linearen Code gibt es eine systematische Codierung.

(b) Zu jedem linearen Code gibt es einen äquivalenten systematischen Code.

Während der Aufwand zur Bestimmung der Minimaldistanz für einen beliebigen Code i.a. mit dem Quadrat der Anzahl der Codeworte wächst, wird er für einen linearen Code beschränkt durch:

(1.7) Beobachtung:

Die Minimaldistanz eines linearen Codes C ist das kleinste Gewicht eines von Null verschiedenen Codewortes, das *Minimalgewicht* von C.

Das Minimalgewicht eines linearen Codes kann mit Hilfe einer Generator- als auch einer Kontrollmatrix ermittelt werden. Wohlbekannt ist:

(1.8) Satz:

Sei C ein (n,k)-Code über GF(q) und H eine Kontrollmatrix von C. Dann besitzt C das Minimalgewicht d genau dann, wenn je d-1 Spalten von H linear unabhängig sind und d linear abhängige Spalten von H existieren.

(1.9) Folgerung:

Das Minimalgewicht eines linearen Codes mit Kontrollmatrix H ist die minimale Anzahl Spalten von H, die zu Null linear kombiniert werden können.

Schließlich sei noch ein Resultat von A.Lempel und S.Winograd angegeben (cf. /Le77/):

(1.10) Satz:

Sei C ein (n,k)-Code mit Minimaldistanz d. Gibt es für alle Paare (w,v) von Codeworten mit $\underline{w} \neq \underline{0}$ und $\underline{v} \neq \underline{0}$ ein $i \in [0:n-1]$, so daß $w_i \neq 0$ und $v_i \neq 0$ gilt, so ist $d \geq k$.

§2 Schranken

Aus der Dimension k und der Minimaldistanz d eines linearen Codes erhält man sofort eine untere Schranke für seine Länge n:

(2.1) Satz (*Singleton-Schranke*):

Für jeden (n,k)-Code mit Minimaldistanz d gilt:

$$n \geq d+k-1.$$

Beweis: Ein systematisch codiertes Codewort mit nur einem von Null verschiedenen Informationskoeffizienten besitzt höchstens das Gewicht n-k+1.

□

(2.2) Definition:

Ein (n,k)-Code für dessen Minimaldistanz

$$d = n-k+1$$

gilt, wird als **MDS-Code** (maximum distance separable), gelegentlich auch als **optimaler Code** bezeichnet.

Triviale MDS-Codes über jedem Grundkörper sind die (n,n-1)-Parity-Check-Codes mit Minimaldistanz d = 2 und die (n,1)-Wiederholungs-Codes mit d = n. Für die Existenz eines nicht-trivialen MDS-Codes über GF(q) mit einer Minimaldistanz $2 < d < n$ muß die Bedingung

$q > k$ erfüllt sein (cf. /He83/); damit existieren insbesondere keine binären, nicht-trivialen MDS-Codes.

Bezeichne $N(k,d,q)$ die minimale Länge eines linearen Codes über $GF(q)$ mit Dimension k und Minimaldistanz d. Eine im Vergleich zur von q unabhängigen Singleton-Schranke deutlich bessere Abschätzung für $N(k,d,q)$ liefert (cf. /Gr60/):

(2.3) **Satz** (Griesmer-Schranke):

$$N(k,d,q) \geq \sum_{i=0}^{k-1} \lceil d/q^i \rceil .$$

Für d von der Form $d = q^{k-1}$ vereinfacht sich Satz (2.3) durch

(2.4) **Beobachtung**:

Für jede natürliche Zahl k gilt

$$\sum_{i=0}^{k-1} \lfloor q^{k-1}/q^i \rfloor = (q^k-1)/(q-1).$$

(2.5) Bemerkung:

Für festes k und hinreichend großes d ist die Griesmer-Schranke scharf, d.h. zu jedem k gibt es ein D(k) derart, daß für alle $d \geq D(k)$ ein linearer (N(k,d,q),k)-Code mit Minimaldistanz d existiert (cf. /Ba73/).

Die Kugeln mit Radius t um die Codewörter eines t-fehlerkorrigierenden Codes müssen disjunkt sein. Diese einfache Beobachtung liefert:

(2.6) Satz (Hamming-Schranke):

Für einen (n,k)-Code über GF(q) mit Minimaldistanz $d = 2t+1$ gilt

$$\sum_{m=0}^{t} \binom{n}{m}(q-1)^m \leq q^{n-k}.$$

(2.7) Definition:

Ein bezüglich der Hamming-Schranke optimaler Code heißt perfekt.

Zahlargumente schließlich führen zu der Schranke (cf. /Be82b/)

(2.8) Satz (*Plotkin-Schranke*):

Für einen (n,k)-Code über GF(q) mit Minimaldistanz d gilt

$$d \cdot (q^k - 1) \leq n \cdot q^{k-1}(q-1).$$

Die bisher betrachteten Schranken dienten der Abschätzung der Länge eines linearen Codes gegebener Dimension und Minimaldistanz nach unten. Abschließend soll noch eine wertvolle obere Schranke für N(k,d,q) angegeben werden (cf. /Ma78/):

(2.9) Satz (*Gilbert-Varshamov-Schranke*):

Es gibt einen (n,k)-Code über GF(q) der Länge n, der mindestens die Dimension k und die Minimaldistanz d besitzt, falls gilt

$$\sum_{m=0}^{d-2} \binom{n-1}{m} (q-1)^m \leq q^{n-k}.$$

§3 Zyklische Codes

(3.1) Definition:

Ein Code C der Länge n **heißt zyklisch**, wenn für jedes Codewort $\underline{c} = (c_0,c_1,\ldots,c_{n-1})$ auch seine zyklische Verschiebung $(c_{n-1},c_0,\ldots,c_{n-2})$ ein Codewort ist.

Das Interesse der Literatur gilt nahezu ausschließlich linearen zyklischen Codes mit der zusätzlichen Einschränkung $(n,q) = 1$ (die in dieser Arbeit als ***zyklische Codes im engeren Sinn*** bezeichnet werden). Im weiteren soll "zyklisch" die Linearität immer einschließen. Dies ermöglicht die Identifizierung des Vektorraums $GF(q)^n$ der Wörter der Länge n mit Komponenten aus $GF(q)$ mit dem Restklassenring $GF(q)[x]/(x^n-1)$, in dem mit Repräsentanten minimalen Grades modulo x^n-1 gerechnet wird. Ein Vektor $\underline{c} = (c_0,c_1,\ldots,c_{n-1})$ kann damit als Polynom

$$c(x) = c_0 + c_1x + \ldots + c_{n-1}x^{n-1}$$

aufgefaßt werden und umgekehrt. Die zyklischen (n,k)-Codes über $GF(q)$ stimmen mit den Idealen in $GF(q)[x]/(x^n-1)$ überein. Aufschluß über die Existenz zyklischer Codes erhält man folglich durch das Zerlegen des Polynoms x^n-1 in irreduzible Faktoren. Ist die Blocklänge n nicht zur Ordnung q teilerfremd, so besitzt der Restklassenring in aller Regel eine wenig übersichtliche Idealstruktur. Die Bedingung $(n,q) = 1$ garantiert, daß das Polynom x^n-1 nur einfache Nullstellen besitzt.

(3.2) Satz:

Sei C ein zyklischer (n,k)-Code über GF(q).

(a) In C gibt es genau ein normiertes Polynom $g(x) \neq 0$ minimalen Grades, das **Generatorpolynom** von C.

(b) Jedes Codewort c(x) von C ist ein Vielfaches von g(x).

Beweis:

(a) Aus der Existenz zweier normierter Codepolynome g(x) und f(x) minimalen Grades r mit $g(x) \neq f(x)$ folgt $g(x)-f(x) \in C$ und $\deg(g(x)-f(x)) < r$ im Widerspruch zur Minimalität von r.

(b) Sei $\deg(g(x)) = r$ und $c(x) \in C$. Dann gibt es eindeutig bestimmte Polynome $q(x), r(x) \in GF(q)[x]/(x^n-1)$ mit

$$c(x) = q(x)g(x) + r(x) \quad \text{und} \quad \deg(r(x)) < r.$$

Wegen $r(x) = c(x) - q(x)g(x)$ ist auch r(x) ein Codewort und somit folgt $r(x) = 0$.

□

Mit der Beweistechnik von Satz (3.2b) kann man weiter zeigen, daß das Generatorpolynom g(x) das Polynom $x^n - 1$ teilt. Umgekehrt ist jeder normierte Teiler von x^n-1 Generatorpolynom eines zyklischen Codes.

(3.3) Beobachtung:

Die Vektoren $g(x)$, $xg(x)$, $x^2g(x)$, ..., $x^{n-r-1}g(x)$ sind linear unabhängig; d.h. für die Dimension k eines zyklischen Codes der Länge n gilt

$$k = n - \deg(g(x)).$$

Die Implementierung der Codierung eines zyklischen Codes beruht meist auf:

(3.4) Beobachtung:

Sei $g(x)$ Generatorpolynom eines zyklischen (n,k)-Codes C über GF(q) und $f(x) \in GF(q)[x]$ mit $\deg(f(x)) < k$. Dann ist

$$c(x) = x^{n-k}f(x) - (x^{n-k}f(x) \bmod g(x))$$

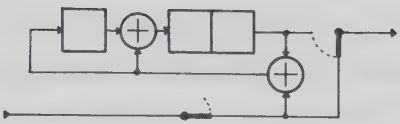
ein Codewort von C.

(3.5) Beispiel:

Der binäre (7,4)-Hamming-Code (cf. /Ma78/) kann durch das Generatorpolynom $g(x) = x^3+x+1$ definiert und mit der in Abbildung (3.6) dargestellten Schaltung seriell codiert werden. Die Codierung ist systematisch und wird durch die Generatormatrix

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

beschrieben.



(3.6) **Abbildung**: Codierung des binären (7,4)-Hamming-Codes

(3.7) **Definition**:

Sei C ein zyklischer (n,k)-Code mit Generatorpolynom g(x). Dann heißt das Polynom $h(x) = (x^n-1)/g(x)$ das ***Kontrollpolynom*** von C.

(3.8) **Bemerkungen**:

(a) Ein Polynom $c(x) \in GF(q)[x]/(x^n-1)$ ist genau dann ein Codewort von C, wenn gilt

$$c(x)\cdot h(x) \equiv 0 \bmod (x^n-1).$$

(b) Der zyklische Code mit Generatorpolynom h(x) ist die Reversion des zu C dualen Codes C^*.

Vielfältiger Nutzen kann aus der Kenntnis der Symmetrie-Eigenschaften eines Codes gezogen werden. Es sei nur an die Bestimmung der Minimaldistanz quadratischer Restklassen-Codes (cf. /Ma78/) und an die Permutations-Decodierverfahren (cf. /Pr62/, /Ma64/) erinnert.

(3.9) Definition:

Ein **Automorphismus** eines Codes der Länge n ist eine Permutation der Indizes |0:n-1| die Codeworte auf Codeworte abbildet.

(3.10) Beobachtungen:

(a) Zyklische Codes sind per definitionem invariant gegen zyklisches Vertauschen der Komponenten ihrer Codeworte, d.h. die Permutation

$$\sigma : i \dashrightarrow i+1 \bmod n$$

ist Automorphismus jedes zyklischen Codes C der Länge n. σ erzeugt eine Automorphismengruppe der Ordnung n.

(b) Ist C ein zyklischer Code der Länge n über GF(q) und gilt (n.q) = 1, so ist die durch

$$\rho : i \dashrightarrow q \cdot i \bmod n$$

definierte Permutation von |0:n-1| ein Automorphismus von C.

Neben der bequemen Beschreibung eines zyklischen Codes mit Hilfe seines Generator- bzw. Kontrollpolynoms besitzt die (mindestens ebenso bequeme) durch seine Nullstellen (bzw. Nicht-Nullstellen) einige Bedeutung.

(3.11) Definition:

Sei g(x) Generatorpolynom eines zyklischen (n,k)-Codes C und α eine primitive n-te Einheitswurzel. α^i heißt *Nullstelle* von C, wenn gilt:

$$g(\alpha^i) = 0 \ .$$

(3.12) Bemerkungen:

(a) Ein zyklischer Code ist durch seine Nullstellen bestimmt; ein Polynom c(x) ist Codewort genau dann, wenn für alle Nullstellen α^i des Codes gilt:

$$c(\alpha^i) = 0.$$

(b) Die Nullstellen des zu einem zyklischen Code C dualen Codes $C^\perp$ sind die Inversen der Nicht-Nullstellen von C.

(3.13) Beobachtung:

Eine Kontrollmatrix H eines zyklischen (n,k)-Codes mit Nullstellenmenge $\{\alpha_0, \alpha_1, \ldots, \alpha_{r-1}\}$ ist gegeben durch

$$H = \begin{bmatrix} 1 & \alpha_0 & \alpha_0^2 & \ldots & \alpha_0^{n-1} \\ 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{n-1} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & \alpha_{r-1} & \alpha_{r-1}^2 & \ldots & \alpha_{r-1}^{n-1} \end{bmatrix}$$

(3.14) Beobachtung:

Sind $g_1(x)$ und $g_2(x)$ Generatorpolynome zyklischer Codes C_1 und C_2, und teilt $g_2(x)$ das Polynom $g_1(x)$, so gilt:

(a) C_1 ist Teilmenge von C_2 und

(b) Die Nullstellenmenge von C_2 ist Teilmenge der Nullstellenmenge von C_1.

(3.15) Definition:

Seien n und q relativ prim und sei j eine ganze Zahl mit $0 \leq j < n$. Die *Kreisteilungsklasse* von j modulo n bezüglich q ist die Menge

$$K(j,q) = \{ j, jq, jq^2, \ldots \pmod n \}.$$

(3.16) Definition:

Das *Minimalpolynom* eines Elementes α von $GF(q^m)$ über $GF(q)$ ist das normierte Polynom kleinsten Grades mit Koeffizienten aus $GF(q)$, das α als Nullstelle besitzt. Zwei Elemente eines Körpers $GF(q^m)$, die dasselbe Minimalpolynom über $GF(q)$ besitzen, heißen *konjugiert* (bezüglich $GF(q)$).

(3.17) Bemerkung:

Die Eigenschaft "konjugiert" definiert eine Äquivalenzrelation. Die Äquivalenzklassen werden *Konjugiertenklassen* oder *Bahnen* genannt.

(3.18) Beobachtungen:

(a) Sei $\alpha \in GF(q^m)$. Dann ist α^q konjugiert zu α.

(b) Die Mächtigkeit einer Konjugiertenklasse teilt den Körpererweiterungsgrad m.

(c) Sei β primitive n-te Einheitswurzel über GF(q), bezeichne $M_j(x)$ das Minimalpolynom von β^j und sei K eine Menge von Repräsentanten der Kreisteilungsklassen modulo n bezüglich q. Dann gilt:

$$M_j(x) = \prod_{i \in K(j,q)} (x-\beta^i)$$

und

$$x^n-1 = \prod_{i \in K} M_i(x) .$$

(d) Sei C zyklischer (n,k)-Code über GF(q) mit Generatorpolynom g(x). Dann gibt es ein $I \subset K$ derart, daß

$$g(x) = \prod_{i \in I} M_i(x) .$$

(3.19) Satz:

Sei C ein zyklischer Code der Länge n über GF(q) mit Generatorpolynom g(x) und gelte (n,q) = 1. Sei ferner α primitive n-te Einheitswurzel über GF(q), bezeichne $M_j(x)$ das Minimalpolynom von α^j und sei J eine Menge von Repräsentanten der Kreisteilungsklassen modulo n bezüglich q derart, daß

$$g(x) = \prod_{j \in J} M_j(x).$$

Dann ist für alle Permutationen

$$\tau : i \dashrightarrow i \cdot t \bmod n \qquad \text{mit } (n,t) = 1$$

das Polynom

$$g'(x) = \prod_{j \in J} M_{\tau(j)}(x)$$

Generatorpolynom eines zu C äquivalenten zyklischen Codes C'.

Beweis: Seien $Z = \bigcup_{j \in J} K(j,q)$ und $Z' = \bigcup_{j \in J} K(\tau(j),q)$. Dann ist jedes Polynom $c(x) \in GF(q)[x]/(x^n - 1)$ mit

$$c(\alpha^i) = 0 \qquad \text{für alle } i \in Z$$

ein Codewort von C. Sei nun $d(x) \in GF(q)[x]/(x^n - 1)$. Offenbar gilt

$$d(x) = \sum_{i=0}^{n-1} d_i x^i = \sum_{i=0}^{n-1} d_{\tau(i)} x^{\tau(i)}.$$

Damit folgt

$d(x) \in C \iff d(\alpha^j) = 0 \qquad$ für alle $j \in Z$

$$\iff \sum_{i=0}^{n-1} d_{\tau(i)} \alpha^{\tau(i)j} = 0 \qquad \text{für alle } j \in Z$$

$$\iff \sum_{i=0}^{n-1} d_{\tau(i)} \alpha^{ij} = 0 \qquad \text{für alle } j \in Z'$$

$$\iff \tau(d(x)) \in C'$$

(3.20) Bemerkungen:

(a) Die Permutationen τ bilden eine zur Einheitengruppe des Rings $\mathbb{Z}_n$ isomorphe Gruppe der Ordnung $\phi(n)$.

(b) Die Permutationen τ mit $t \in K(1,q)$ bilden eine Untergruppe der Automorphismengruppe eines zyklischen Codes (vgl. Beobachtung (3.10b)).

(3.21) Definition:

Eine Menge M von n-ten Einheitswurzeln in einem Körper $GF(q^m)$ heißt **konsekutiv**, wenn für eine primitive n-te Einheitswurzel β und eine natürliche Zahl b gilt:

$$M = \{ \beta^b, \beta^{b+1}, \ldots, \beta^{b+k} \}.$$

Mit Hilfe der konsekutiven Nullstellen eines zyklischen Codes C kann seine Minimaldistanz leicht abgeschätzt werden.

(3.22) Bemerkung:

Enthält die Nullstellenmenge eines zyklischen Codes C eine konsekutive Teilmenge der Mächtigkeit d^*, so gilt für die Minimaldistanz d des Codes:

$$d \geq d^*+1$$ (**BCH-Schranke**, vgl. /Ma78/).

Enthält die Nullstellenmenge von C mehrere konsekutive Teilmengen, so kann die obige Abschätzung verbessert werden (vgl. /Ha72a/, /Ro82/, /Ro83/).

§4 Code-Modifikationen

In der Literatur wird eine Vielzahl von Möglichkeiten zur Modifikation eines gegebenen Codes beschrieben (siehe z.B. /Ma78/, /Bl83/, /He83/, /Fu84/). Gemeinsames Merkmal der hier betrachteten Code-Modifikationen ist die Veränderung des zugrundeliegenden Alphabets.

(4.1) Beobachtung:

Sei C ein (n,k)-Code über $GF(q^m)$ mit Generatormatrix G und Minimaldistanz d und sei $B = <\beta_0,\beta_1,\ldots,\beta_{m-1}>$ eine Basis von $GF(q^m)$ über GF(q). Repräsentiert man die Komponenten der Codewörter von C als Vektoren der Länge m über GF(q) bezüglich der Basis B, so erhält man einen $(n\cdot m, k\cdot m)$-Code C' über GF(q) mit Minimaldistanz $d' \geq d$ und die Matrix

$$G' = \begin{bmatrix} \beta_0 \cdot G \\ \beta_1 \cdot G \\ \ldots \\ \beta_{m-1} \cdot G \end{bmatrix}$$

ist eine Generatormatrix von C'.

(4.2) **Bemerkung**:

Das Bild eines zyklischen Codes unter der Modifikation (4.1) ist im allgemeinen kein zyklischer Code. Gewichtsverteilung und Minimaldistanz des resultierenden Codes C' sind von der zugrundeliegenden Basis B abhängig.

(4.3) **Definition**:

Sei C ein (n,k)-Code über $GF(q^m)$ mit Minimaldistanz d. Dann heißt

$$tr(C) = \{ (tr(c_0),tr(c_1),\ldots,tr(c_{n-1})) \mid (c_0,c_1,\ldots,c_{n-1}) \in C \}$$

der **Spur-Code** von C (zu Definition und Eigenschaften der Spurabbildung siehe auch I.1.6 ff).

(4.4) **Definition**:

Sei C ein Code der Länge n über $GF(q^m)$ und sei

$C|GF(q) = \{ \underline{c}=(c_0,c_1,\ldots,c_{n-1}) \in C \mid c_i \in GF(q) \text{ für } 0 \leq i < n \}$.

Der Code $C|GF(q)$ heißt die ***Schrumpfung*** von C.

(4.5) **Bemerkungen** (cf. /De75/, /Ma78/):

(a) $tr(C)$ ist ein linearer (n,k')-Code über $GF(q)$ mit Minimaldistanz $d' \geq d$ und $k \leq k' \leq m \cdot k$.

(b) Die Schrumpfung $C' = C|GF(q)$ eines linearen (n,k)-Codes C über $GF(q^m)$ mit Minimaldistanz d ist ein

linearer (n,k')-Code mit Minimaldistanz d', wobei gilt

$$d' \geq d \quad \text{und}$$

$$\max(0, n-m(n-k)) \leq k' \leq k \ .$$

(c) $(C|GF(q))^{\perp} = tr(C^{\perp})$.

(d) Die Schrumpfung eines zyklischen (n,k)-Codes mit der Nullstellenmenge $\{ \alpha^i \mid i \in I \}$ ist ein zyklischer (n,k')-Code mit der Nullstellenmenge

$$\{ \alpha^j \mid j \in \cup_{i \in I} K(i,q) \}.$$

(4.6) **Beispiel**:

Sei $q = 2^6$, $n = 9$ und $\alpha \in GF(2^6)$ primitive 9-te Einheitswurzel. Der durch das Generatorpolynom

$$g(x) = (x-\alpha)(x-\alpha^2)$$

definierte zyklische Code C über $GF(2^6)$ besitzt die Dimension $k = 7$ und die Minimaldistanz $d = 3$ (vgl. Bemerkung (3.22)).

$$\{\alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^7, \alpha^5\}$$

ist die Nullstellenmenge der Schrumpfung $C' = C|GF(2)$ mit Dimension $k' = 3$ und Minimaldistanz $d' = 3$ (vgl. Anhang (A3)). Wegen $tr(C) = (C^{\perp}|GF(q))^{\perp}$ besitzt der Spur-Code von C die Dimension $k'' = 9$ und die Minimaldistanz $d'' = 1$.

§5 BCH-Codes, Reed-Solomon-Codes und Verallgemeinerungen

Die Klasse der von A.Hocquenghem /Ho59/ und unabhängig von R.C.Bose und D.K.Ray-Chaudhuri /Bo60/ gefundenen Codes ist die wohl wichtigste und bestuntersuchte Teilklasse zyklischer Codes. BCH-Codes sind einfach zu konstruieren und besitzen für nicht zu große Länge gute Fehlerkorrektureigenschaften (cf. /Ma78/). Zudem sind effiziente Decodierverfahren bekannt (cf. /Be68/, /Bl83/).

Die nach I.S.Reed und G.Solomon /Re60/ benannte Teilklasse der BCH-Codes hat in der Praxis zunehmend Bedeutung erlangt. So basiert z.B. das bei der 'compact disc' eingesetzte Fehlerkorrekturverfahren auf der Verschachtelung zweier verkürzter Reed-Solomon-Codes über dem Körper $GF(2^8)$ (cf. /Ca82/, /Ho82/).

(5.1) Definition:

Ein zyklischer Code der Länge n über GF(q) mit Generatorpolynom g(x) ist ein *BCH-Code zur Entwurfsdistanz d'*, falls für ein $b \geq 0$ gilt:

$$g(x) = kgV \{ M_b(x), M_{b+1}(x), \ldots, M_{b+d'-2}(x) \}.$$

BCH-Codes mit b = 1 heißen *BCH-Codes im engeren Sinn*. BCH-Codes der Länge $n = q^m-1$ heißen *primitiv*.

(5.2) **Beispiel**:

Die primitiven BCH-Codes im engeren Sinn über GF(2) mit $g(x) = M_1(x)$ sind $(2^m-1, 2^m-m-1)$-Codes mit Minimaldistanz $d = 3$, die binären *Hamming-Codes*. Hamming-Codes sind perfekt (diese Codes werden von einigen Autoren auch als Golay-Hamming-Codes bezeichnet; zur Auseinandersetzung um die "Vaterschaft" sei die Lektüre von /Th83/ empfohlen).

(5.3) **Definition**:

Ein *Reed-Solomon-Code* über GF(q) ist ein primitiver BCH-Code mit $m = 1$.

(5.4) **Bemerkung**:

Das Generatorpolynom g(x) eines Reed-Solomon-Codes der Entwurfsdistanz d' ist von der Form

$$g(x) = (x-\alpha^b)(x-\alpha^{b+1})\ldots(x-\alpha^{b+d'-2}).$$

Für die Dimension k folgt

$$k = n - \deg(g(x)) = n - d' + 1.$$

Die BCH-Schranke garantiert eine Minimaldistanz $d \geq d'$. Aufgrund der Singleton-Schranke (Satz (2.1)) muß die Gleichheit gelten. Reed-Solomon-Codes sind folglich (nicht-triviale) MDS-Codes.

(5.5) **Bemerkung** (cf. /Bl83/):

Jeder BCH-Code ist die Schrumpfung eines Reed-Solomon-Codes.

(5.6) **Definition**:

Seien $\alpha = (\alpha_0,\alpha_1,\ldots,\alpha_{n-1})$ mit $\alpha_i \neq 0$, $\alpha_i \neq \alpha_j$ für $i \neq j$ und $\nu = (\nu_0,\nu_1,\ldots,\nu_{n-1})$ mit $\nu_i \neq 0$ Vektoren über $GF(q^m)$. Dann ist

$$C = \{ (\nu_0 f(\alpha_0),\nu_1 f(\alpha_1),\ldots,\nu_{n-1} f(\alpha_{n-1})) \mid f(x) \in GF(q^m)[x] \text{ und } \deg(f(x)) < k \}$$

ein (n,k)-Code über $GF(q^m)$, der als *verallgemeinerter Reed-Solomon-Code* bezeichnet wird.

(5.7) **Beobachtungen**:

(a) Reed-Solomon-Codes sind verallgemeinerte Reed-Solomon-Codes mit $n = q^m-1$ und $\nu = \underline{1}$.

(b) Da das Polynom f(x) höchstens k-1 Nullstellen besitzt, folgt für die Minimaldistanz eines verallgemeinerten Reed Solomon-Codes $d \geq n-(k-1)$ und mit der Singleton-Schranke $d = n-k+1$, d.h. verallgemeinerte Reed-Solomon-Codes sind ebenfalls MDS-Codes.

Verschiedene Verallgemeinerungen der Klasse der BCH-Codes stehen in enger Beziehung zur Klasse der verallgemeinerten Reed-Solomon-Codes:

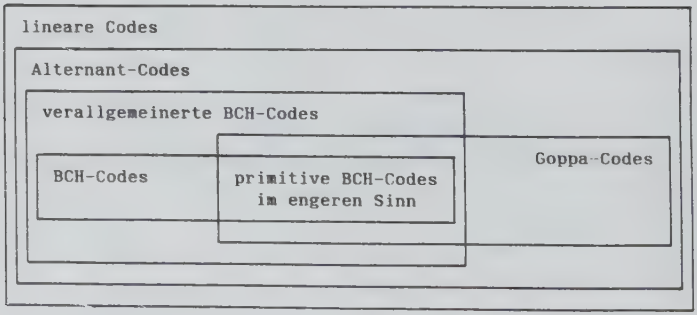
(5.8) Bemerkungen (cf. /Ma78/):

(a) Die von H.J.Helgert /He72/ gefundenen Alternant-Codes erhält man durch Schrumpfung aus den verallgemeinerten Reed-Solomon-Codes.

(b) Die Klasse der nach V.D.Goppa /Go70/ benannten linearen Codes kann als interessanteste Teilklasse der Alternant-Codes somit ebenfalls durch Schrumpfung verallgemeinerter Reed-Solomon-Codes erhalten werden.

(c) Schließlich ist auch die Klasse der von R.T.Chien und D.M.Choy verallgemeinerten BCH-Codes /Ch75/ eine Teilklasse der Alternant-Codes.

In Abbildung (5.9) sind die Teilmengenbeziehungen wichtiger Codeklassen dargestellt.



(5.9) Abbildung: Codeklassen und ihre Teilmengenbeziehungen

I. Arithmetik in endlichen Körpern

Algebraische Decodierverfahren für über einem Körper GF(q) definierte Codes (wie z.B. der Berlekamp-Algorithmus zur Decodierung von BCH-Codes) benötigen in der Regel Arithmetik in einem Erweiterungskörper $GF(q^m)$ über GF(q). Zur Hardware-Implementierung derartiger Arithmetik finden sich in der Literatur verschiedene Vorschläge (cf. /Ba63/, /Be68/, /Ta68/, /Pe72/, /Ma78/, /Wi80/), die fast ausnahmslos (und nicht selten stillschweigend) von der Darstellung von $GF(q^m)$ bzgl. einer polynomialen Basis über GF(q) ausgehen.

Die in den Paragraphen 2 und 3 vorgestellten Implementierungen verschiedener Verfahren zur Multiplikation und zur Berechnung der multiplikativen Inversen (die Addition in $GF(q^m)$ kann unabhängig von der Basisdarstellung auf m GF(q)-Additionen zurückgeführt werden) beruhen zumeist nicht auf der genannten Standard-Darstellung. Wegen ihrer regulären Struktur werden speziell die parallelen Versionen der auf einer Normalbasis-Darstellung basierenden arithmetischen Operationen als für VLSI-Implementierungen besonders geeignet angesehen. Rein auf die speziellen Anforderungen des VLSI-Designs ausgerichtete $GF(2^m)$-Multiplizierer werden z.B. in /La71/ und /Ye84/ beschrieben.

§1 Basen endlicher Körper

Ein Erweiterungskörper $GF(q^m)$ eines endlichen Körpers $GF(q)$ kann als m-dimensionaler Vektorraum über $GF(q)$ aufgefaßt werden. Von der im allgemeinen sehr großen Anzahl verschiedener Basen von $GF(q^m):GF(q)$ (cf. /Co68/) sind für die Implementierung arithmetischer Operationen in $GF(q^m)$ vor allem zwei spezielle Klassen, polynomiale Basen und Normalbasen, von Bedeutung.

(1.1) Definition:

Eine Basis von $GF(q^m)$ über $GF(q)$ der Form $<1,\alpha,\alpha^2,\ldots,\alpha^{m-1}>$ heißt eine *polynomiale Basis* von $GF(q^m)$ über $GF(q)$.

(1.2) Bemerkung:

Für ein $\alpha \in GF(q^m)$ mit Minimalpolynom $M(x) \in GF(q)[x]$ und $\deg(M(x)) = m$ ist $<1,\alpha,\alpha^2,\ldots,\alpha^{m-1}>$ eine (polynomiale) Basis in $GF(q^m)$ über $GF(q)$.

(1.3) Definition:

Eine von den Elementen einer Konjugiertenklasse gebildete Basis von $GF(q^m)$ über $GF(q)$ heißt eine *Normalbasis* von $GF(q^m)$ über $GF(q)$.

(1.4) Beobachtung:

Eine GF(q)-Basis $<\beta_0,\beta_1,\ldots,\beta_{m-1}>$ von $GF(q^m)$ ist eine Normalbasis genau dann, wenn die Basiselemente β_i durch den Körper-Automorphismus

$$\sigma : \beta_i \dashrightarrow \beta_i^q$$

permutiert werden.

(1.5) Bemerkungen:

(a) In jedem Körper $GF(q^m)$ gibt es eine Normalbasis über GF(q).

(b) In jedem Körper $GF(q^m)$ gibt es ein primitives Element α derart, daß die Elemente der Konjugiertenklasse von α eine Normalbasis von $GF(q^m)$ über GF(q) bilden (Satz von Davenport, cf. /Da68/).

(1.6) Definition:

Für ein Element α aus $GF(q^m)$ ist die Spur $tr(\alpha)$ über GF(q) definiert durch

$$tr(\alpha) = \sum_{i=0}^{m-1} \alpha^{q^i} .$$

Die im folgenden angegebenen Eigenschaften der Spur sind wohlbekannt (cf. /Li83/).

(1.7) **Bemerkungen**:

Seien $\alpha, \beta \in GF(q^m)$ und sei $c \in GF(q)$. Dann gilt

(a) $tr(\alpha+\beta) = tr(\alpha) + tr(\beta)$

(b) $tr(c\alpha) = c \cdot tr(\alpha)$

(c) $tr(c) = m \cdot c$

(d) $tr(\alpha^q) = tr(\alpha)$

(e) $tr(\alpha) \in GF(q)$

(1.8) **Bemerkungen**:

(a) Für einen Körper $GF(q^m)$ nimmt die Spur über $GF(q)$ jeden Wert von $GF(q)$ gleich oft, d.h. q^{m-1}-mal an.

(b) Jede lineare Abbildung von $GF(q^m)$ nach $GF(q)$ ist von der Form $tr(\alpha x)$ für ein $\alpha \in GF(q^m)$ (cf. /Co68/).

Die Spur von $\alpha \in GF(q^m)$ über $GF(q)$ kann direkt aus dem charakteristischen Polynom von α abgelesen werden:

(1.9) **Definition**:

Sei $\alpha \in GF(q^m)$, $M(x)$ das Minimalpolynom von α über $GF(q)$ und $d = \deg(M(x))$. Dann teilt d den Erweiterungsgrad m und $G(x) = M(x)^{m/d}$ heißt das *charakteristische Polynom* von α über $GF(q)$.

(1.10) Bemerkungen:

(a) Die Nullstellen von G(x) sind genau die Konjugierten von α.

(b) Sei $G(x) = x^m + G_{m-1}x^{m-1} + \ldots + G_1x + G_0$. Dann gilt

$$tr(\alpha) = -G_{m-1}.$$

Sei $\alpha \in GF(q^m)$, G(x) das charakteristische Polynom von α über GF(q) und bezeichne $G_i(x)$ das charakteristische Polynom von α^i. Zur Berechnung der Polynome $G_i(x)$ aus G(x) sind mehrere Verfahren bekannt (cf. /Li83/). Die hier skizzierte Methode beruht auf der Beziehung zwischen dem Polynom G(x) und seiner Begleitmatrix.

(1.11) Definition:

Sei $f(x) = x^m + f_{m-1}x^{m-1} + \ldots + f_1x + f_0$ ein normiertes Polynom $\in GF(q)[x]$. Dann heißt die (m×m)-Matrix

$$F = ((F_{ij})) = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 & -f_0 \\ 1 & 0 & 0 & \ldots & 0 & 0 & -f_1 \\ 0 & 1 & 0 & \ldots & 0 & 0 & -f_2 \\ 0 & 0 & 1 & \ldots & 0 & 0 & -f_3 \\ \cdot & \cdot & \cdot & & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \ldots & 0 & 1 & -f_{m-1} \end{bmatrix}$$

die *Begleitmatrix* von f(x).

(1.12) Bemerkung:

Das charakteristische Polynom $G(x)$ von $\alpha \in GF(q^m)$ über $GF(q)$ ist auch charakteristisches Polynom seiner Begleitmatrix F, d.h. es gilt

$$G(x) = \det(xI_m - F)$$

und die charakteristischen Polynome $G_i(x)$ der α^i können mit Hilfe der Potenzen von F berechnet werden (cf. /Li83/):

$$G_i(x) = \det(xI_m - F^i).$$

Für die Spur von α^i folgt hieraus insbesondere

$$tr(\alpha^i) = \sum_{j=0}^{m-1} (F^i)_{jj}.$$

(1.13) Definition:

Zwei Basen $A = <\alpha_0,\alpha_1,\ldots,\alpha_{m-1}>$ und $B = <\beta_0,\beta_1,\ldots,\beta_{m-1}>$ heißen (spur-) ***dual*** (auch ***orthogonal***), wenn gilt:

$tr(\alpha_i\beta_i) = 1$ für $i \in [0:m-1]$

$tr(\alpha_i\beta_j) = 0$ für $i,j \in [0:m-1]$ mit $i \neq j$.

A und B heißen ***schwach spurdual***, wenn eine Permutation P von $[0:m-1]$ existiert, so daß

$tr(\alpha_i\beta_{P(i)}) = 1$ für $i \in [0:m-1]$

$tr(\alpha_i\beta_{P(j)}) = 0$ für $i,j \in [0:m-1]$ mit $i \neq j$.

(1.14) **Bemerkungen**:

(a) Zu jeder GF(q) Basis B in $GF(q^m)$ gibt es eine (eindeutig bestimmte) duale Basis.

(b) Ist B eine Normalbasis, so ist auch die zu B orthogonale Basis Normalbasis.

(c) Ist B eine polynomiale Basis, so ist die zu B orthogonale Basis i.a. keine polynomiale Basis.

Das folgende Lemma stellt eine zur Herleitung effizienter Multiplikationsverfahren ausnutzbare Verbindung zwischen dualen Basen her (siehe §2).

(1.15) **Lemma**:

Sei $A = \langle \alpha_0, \alpha_1, \ldots, \alpha_{m-1} \rangle$ eine GF(q)-Basis in $GF(q^m)$ und $B = \langle \beta_0, \beta_1, \ldots, \beta_{m-1} \rangle$ eine zu A spur-duale Basis. Für $w \in GF(q^m)$ gilt

$$w = \sum_{i=0}^{m-1} tr(w\alpha_i)\beta_i \ .$$

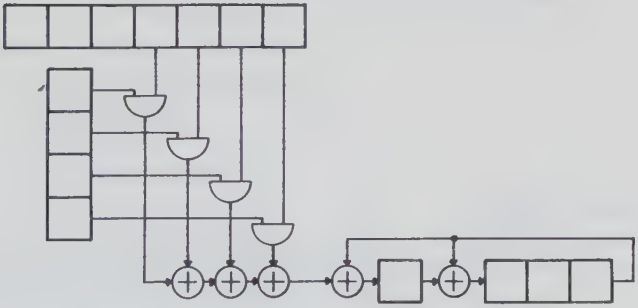
Beweis: $$\mathrm{tr}(w\alpha_j) = \mathrm{tr}\Big(\alpha_j \sum_{i=0}^{m-1} w_i\beta_i\Big)$$

$$= \sum_{i=0}^{m-1} \mathrm{tr}(\alpha_j w_i \beta_i)$$

$$= \sum_{i=0}^{m-1} w_i(\mathrm{tr}(\alpha_j\beta_i)) = w_j$$

§2 Multiplikationsverfahren

Die Multiplikation zweier in polynomialer Basis gegebener $GF(q^m)$-Elemente entspricht der zyklischen Faltung ihrer assoziierten Polynome. Sie kann bei zu m proportionalem Implementierungsaufwand in 2m-1 Takten durchgeführt werden. Abbildung (2.1) stellt einen derartigen Multiplizierer für die Parameter q = 2, m = 4 und die Basis $<1,\alpha,\alpha^2,\alpha^3>$ (α sei Nullstelle des irreduziblen Polynoms x^4+x+1) dar.



(2.1) Abbildung: Multiplikation durch Faltung

Die im folgenden vorgestellten Multiplizierer zeigen, daß insbesondere bei geeigneter Wahl der Basis von $GF(q^m)$ über GF(q) günstigere Realisierungen möglich sind.

(2.2) Beobachtung:

Sei $B = \langle \beta_0, \beta_1, \ldots \beta_{m-1} \rangle$ Normalbasis von $GF(q^m)$ über $GF(q)$ mit $\beta_{i+1} = \beta_i^q$ und sei das Produkt

$$v \cdot w = \sum_{i=0}^{m-1} u_i \beta_i$$

der in der Basis B gegebenen Operanden v und w zu bestimmen.

Sei ferner

$$u_i = f(v,w) = \sum_{j=0}^{m-1} f_j(v_j, w).$$

Dann gilt

$$u_{i-1} = f(v^q, w^q) = \sum_{j=0}^{m-1} f_j(v_{j-1}, w^q).$$

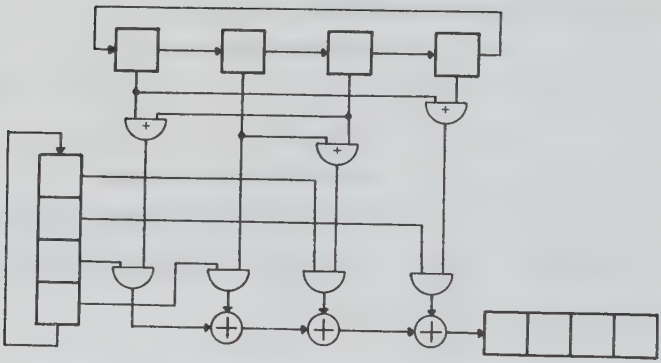
(2.3) Beispiel:

Sei $q = 2$, $m = 4$ und α Nullstelle des irreduziblen Polynoms $x^4 + x + 1 \in GF(2)[x]$. Dann ist $B = \langle \alpha^3, \alpha^6, \alpha^{12}, \alpha^9 \rangle$ eine Normalbasis von GF(16) über GF(2) und es gilt

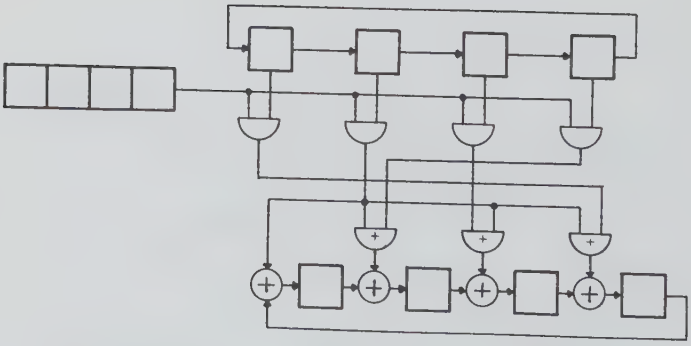
$$u_0 = v_0w_2 + v_1(w_3+w_2) + v_2(w_0+w_1) + v_3(w_1+w_3)$$
$$u_1 = v_0(w_2+w_0) + v_1w_3 + v_2(w_0+w_3) + v_3(w_1+w_2)$$
$$u_2 = v_0(w_2+w_3) + v_1(w_3+w_1) + v_2w_0 + v_3(w_1+w_0)$$
$$u_3 = v_0(w_2+w_1) + v_1(w_3+w_0) + v_2(w_0+w_2) + v_3w_1 \quad .$$

Mit Beobachtung (2.2) können zwei unterschiedliche serielle Normalbasis-Multiplizierer realisiert werden, die es gestatten, das Produkt $u = v \cdot w$ in m Takten zu berechnen. Der Multiplizierer vom Typ A arbeitet hierbei i-sequentiell, d.h. er ermittelt (beginnend mit u_{m-1}) sukzessive die Koeffizienten des Ergebnisses. Der Multiplizierer vom Typ B dagegen arbeitet j-sequentiell, d.h. er benötigt einen der Operanden (beginnend mit dem höchstwertigen Koeffizienten) seriell. Im Implementierungsaufwand unterscheiden sich die beiden Multiplizierer nicht.

In den Abbildungen (2.4) und (2.5) sind die aus Beispiel (2.3) resultierenden Implementierungen der Multiplikation in GF(16) dargestellt.



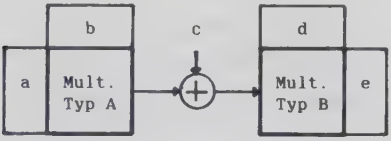
(2.4) **Abbildung**: Serieller Normalbasis-Multiplizierer vom Typ A



(2.5) **Abbildung**: Serieller Normalbasis-Multiplizierer vom Typ B

(2.6) Bemerkungen:

(a) Durch geeignete Zusammenschaltung von Multiplizierern der beiden Typen ist auch die Ausführung aufeinanderfolgender Multiplikationen in m Takten möglich. Beispielsweise kann die für die Iterationen des Berlekamp-Algorithmus zur Decodierung von BCH-Codes (cf. /Be68/) typische Operation e := (ab+c)·d mit der dargestellten Anordnung in m Takten berechnet werden.



(b) Für in polynomialer Basis vorliegende Operanden können ebenfalls Multiplizierer vom Typ B realisiert werden (cf. /Be68/, /Ma78/).

Lemma (1.15) ermöglicht die Implementierung serieller Multiplizierer vom Typ A wie vom Typ B für Operanden v und w, falls v in polynomialer Basis und w in dazu dualer Basis (in der auch das Produkt dargestellt wird) gegeben ist (cf. /Hs84/, /Be82a/).

(2.7) **Beobachtung**:

Sei $A = \langle 1, \alpha, \alpha^2, \ldots, \alpha^{m-1} \rangle$ eine polynomiale GF(q)-Basis in $GF(q^m)$ und $B = \langle \beta_0, \beta_1, \ldots, \beta_{m-1} \rangle$ eine zu A spur-duale Basis. Seien

$$w = \sum_{i=0}^{m-1} w_i \beta_i \quad , \qquad v = \sum_{i=0}^{m-1} v_i \alpha^i$$

und sei das Produkt

$$v \cdot w = \sum_{i=0}^{m-1} u_i \beta_i$$

zu bestimmen. Für $0 \leq i \leq m-1$ gilt

$$u_i = tr(vw\alpha^i)$$

$$= \sum_{j=0}^{m-1} tr(v_j w \alpha^{i+j})$$

$$= \sum_{j=0}^{m-1} v_j (tr(w\alpha^{i+j}))$$

$$= \sum_{j=0}^{m-1} v_j (w\alpha^i)_j ,$$

d.h. u_i kann im Fall q = 2 mittels einer (i-sequentiellen) Schaltung aus m AND-Gattern und m-1 XOR-Gattern in $O(\log_2(m))$ Gatterlaufzeiten aus v und $w\alpha^i$ berechnet werden.

Die Multiplikation des Elements w mit α kann dabei wegen

$$w\alpha = \sum_{i=0}^{m-1} tr(w\alpha^{i+1})\beta_i = \sum_{i=0}^{m-2} w_{i+1}\beta_i + tr(w\alpha^m)\beta_{m-1}$$

in einem Takt mit einem Rückkopplungs-Schieberegister durchgeführt werden. Die j-sequentielle Berechnung des Produkts von v und w ist mit gleichem Aufwand implementierbar.

(2.8) Beispiel:

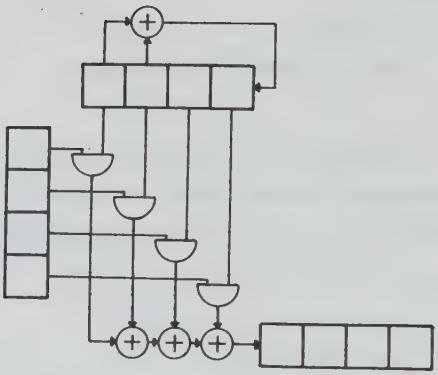
Sei q = 2, m = 4 und α Nullstelle des irreduziblen Polynoms $x^4 + x + 1 \in GF(2)[x]$. Dann ist $A = \langle 1,\alpha,\alpha^2,\alpha^3\rangle$ eine polynomiale Basis von GF(16) über GF(2), $B = \langle \alpha^{14},\alpha^2,\alpha,1\rangle$ zu A orthogonale Basis und es gilt

$$(w\alpha)_3 = tr(w\alpha^4) = w_0 + w_1 ,$$

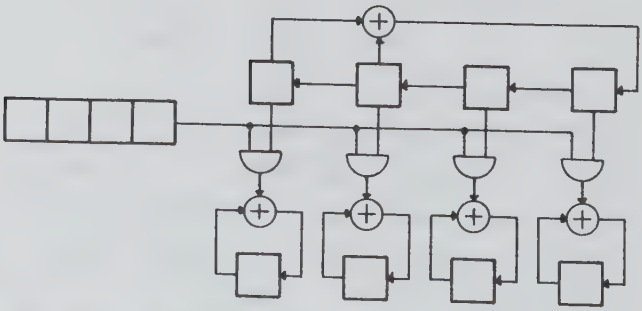
sowie

$$\begin{aligned}
u_0 &= v_0w_0 + v_1w_1 + v_2w_2 + v_3w_3\\
u_1 &= v_0w_1 + v_1w_2 + v_2w_3 + v_3(w_0+w_1)\\
u_2 &= v_0w_2 + v_1w_3 + v_2(w_0+w_1) + v_3(w_1+w_2)\\
u_3 &= v_0w_3 + v_1(w_0+w_1) + v_2(w_1+w_2) + v_3(w_2+w_3) .
\end{aligned}$$

Die aus Beispiel (2.8) resultierenden Implementierungen der Multiplikation in GF(16) sind in den Abbildungen (2.9) und (2.10) dargestellt.



(2.9) **Abbildung**: Serieller Multiplizierer vom Typ A



(2.10) **Abbildung**: Serieller Multiplizierer vom Typ B

Für die Implementierung der Multiplikation zweier Variabler nach Beobachtung (2.7) wäre die Existenz selbst-dualer oder schwach selbst-dualer polynomialer Basen wünschenswert. Im folgenden wird gezeigt, daß es in dem für Anwendungen besonders wichtigen Fall $q = 2$ selbst-duale polynomiale Basen nicht geben kann. Für gewisse Erweiterungsgrade existieren jedoch schwach selbst-duale polynomiale Basen.

(2.11) Lemma:

Für $m \geq 3$ existiert keine selbst-duale polynomiale GF(q)-Basis in $GF(q^m)$.

Beweis: Angenommen, $<1,\alpha,\ldots,\alpha^{m-1}>$ wäre selbst-duale polynomiale GF(q)-Basis in $GF(q^m)$. Daraus folgte

$$0 = tr(\alpha^{2+0}) = tr(\alpha^{1+1}) = 1.$$

□

(2.12) Satz:

Es existiert keine selbst-duale polynomiale GF(2)-Basis in $GF(2^m)$.

Beweis: Wegen Lemma (2.11) ist nur noch der Fall $m = 2$ zu zeigen. Angenommen, $<1,\alpha>$ wäre selbst-duale polynomiale GF(2)-Basis

in GF(4). Daraus folgte

$$0 = tr(\alpha) = tr(\alpha^2) = tr(\alpha^{1+1}) = 1.$$

□

(2.13) Lemma:

Für $m \equiv 0 \bmod 2$ existiert keine schwach selbst-duale polynomiale GF(2)-Basis in $GF(2^m)$.

Beweis: Angenommen, $<1,\alpha,\ldots,\alpha^{m-1}>$ ist eine schwach selbst-duale polynomiale GF(2)-Basis in $GF(2^m)$ und $m \equiv 0 \bmod 2$. Dann ist $tr(1) = 0$ und für genau ein $i \in [1:m-1]$ gilt $tr(\alpha^i) = 1$. Daraus folgt $tr(\alpha^{2i}) = 1$ im Widerspruch zur Annahme.

□

(2.14) Satz:

Eine schwach selbst-duale polynomiale GF(2)-Basis in $GF(2^m)$ existiert genau dann, wenn das Polynom $x^m + x + 1$ irreduzibel über GF(2) und $m \equiv 1 \bmod 2$ ist.

Beweis:

"==>" Sei $<1,\alpha,\alpha^2,\ldots,\alpha^{m-1}>$ schwach selbst-duale polynomiale GF(2)-Basis in $GF(2^m)$. Wegen Lemma (2.13) folgt $m \equiv 1 \bmod 2$ und somit

$$tr(1) = 1,$$

$$tr(\alpha^i) = 0 \quad \text{für} \quad 1 \leq i \leq m-1,$$

$$tr(\alpha^m) = 1 \quad \text{und}$$

(*) $$tr(\alpha^i) = 0 \quad \text{für} \quad m+1 \leq i \leq 2m-1.$$

Damit gilt

$$tr(\alpha^i) = (\alpha^i)_0$$

und mit (*) folgt, daß das Minimalpolynom von α von der Form

$$x^m + x + 1$$

ist.

"<==" Sei $m \equiv 1 \bmod 2$ und $f(x) = x^m + x + 1$ Minimalpolynom von $\alpha \in GF(2^m)$. Dann ist $f(x)$ charakteristisches Polynom von α über GF(2) mit der Begleitmatrix

$$F = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 & 1 \\ 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 & 0 \\ \vdots & & & & & \vdots & \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix}.$$

Aus Bemerkung (1.12) folgt sofort

$$tr(\alpha^i) = 0 \quad \text{für} \quad 1 \leq i \leq m-1$$

und mit $tr(1) = 1$

$$tr(\alpha^i) = (\alpha^i)_0.$$

Damit ist $<1, \alpha^{m-1}, \alpha^{m-2}, \dots, \alpha>$ dual zur polynomialen Basis $<1, \alpha, \alpha^2, \dots, \alpha^{m-1}>$.

□

Die Irreduzibilität der Trinome vom Typ $x^m + x + 1$ über GF(2) wurde für $2 \leq m \leq 30000$ mit Hilfe einer Großrechenanlage untersucht (cf. /Zi70/). Die 16 Werte m in diesem Bereich, für die $m \equiv 1 \bmod 2$ gilt und das Polynom $x^m + x + 1$ irreduzibel über GF(2) ist, sind in Tabelle (2.15) aufgeführt. Soweit bekannt, ist daneben angegeben, ob es sich um ein primitives Polynom handelt oder nicht.

m		m	
3	primitiv	471	
7	primitiv	865	
9	nicht primitiv	4495	
15	primitiv	6321	
63	primitiv	7447	
127	primitiv	11425	
153	primitiv	24369	
303		28713	

(2.15) Tabelle: Über GF(2) irreduzible Trinome $x^m + x + 1$

Auch bei nicht schwach selbst-dualen Basen ist die Transformation eines $GF(2^m)$-Elementes aus seiner Darstellung in polynomialer Basis in die dazu orthogonale Basis und umgekehrt oft mit geringem Aufwand implementierbar. So gilt z.B.:

(2.16) Satz:

Sei $\alpha \in GF(2^m)$ mit Minimalpolynom $x^m + x + 1$ und sei $m \equiv 0 \bmod 2$. Dann ist $<\alpha^{-1}=1+\alpha^{m-1},\alpha^{m-2},\alpha^{m-3},\ldots,1>$ duale Basis zur polynomialen Basis $<1,\alpha,\alpha^2,\ldots,\alpha^{m-1}>$.

Beweis: Analog zum Beweis von Satz (2.14) folgt $tr(\alpha^i) = (\alpha^i)_{m-1}$ und daraus die Behauptung.

□

In diesem Fall können die Basistransformationen mit je einem XOR-Gatter realisiert werden. In Tabelle (2.17) sind analog zur Tabelle (2.15) diejenigen Werte m mit $2 \leq m \leq 30000$ aufgeführt, für die $m \equiv 0 \bmod 2$ gilt und das Polynom $x^m + x + 1$ irreduzibel über GF(2) ist (cf. /Zi70/).

m		m	
2	primitiv	532	primitiv
4	primitiv	900	
6	primitiv	1366	
22	primitiv	2380	
28	nicht primitiv	3310	
30	nicht primitiv	10198	
46	nicht primitiv	21846	
60	primitiv	27286	
172	nicht primitiv		

(2.17) Tabelle: Über GF(2) irreduzible Trinome $x^m + x + 1$

Tabelle (2.18) schließlich ist das Resultat einer Computersuche und gibt für die Erweiterungsgrade $2 \leq m \leq 12$ die für die Transformation zwischen der durch konsekutive Potenzen einer Nullstelle des angegebenen Minimalpolynoms definierten polynomialen GF(2)-Basis über $GF(2^m)$ und ihrer dualen Basis ausreichende Anzahl an GF(2)-Additionen an.

m	Minimalpolynom	#
2	(111)	1
3	(1101)	0
4	(11001)	1
5	(100101)	2
6	(1100001)	1
7	(11000001)	0
8	(111000011)	9
9	(1100000001)	0
10	(10010000001)	3
11	(1101010000001)	5
12	(1110100000001)	5

(2.18) Tabelle

Für viele Anwendungen ist die Implementierung der Multiplikation mit Konstanten ausreichend. In diesen Fällen können wesentlich einfachere serielle Multiplizierer eingesetzt werden (cf. /Be68/, /Ma78/). Die auf einer Normalbasis-Darstellung basierende parallele Implementierung der Multiplikation mit Konstanten ist aufgrund ihrer Modularität für VLSI-Implementierungen besonders interessant.

(2.19) Beobachtung:

Sei $B = \langle \beta_0, \beta_1, \ldots \beta_{m-1} \rangle$ Normalbasis von $GF(q^m)$ über $GF(q)$ mit $\beta_{i+1} = \beta_i^q$ und sei das Produkt

$$v \cdot \delta = \sum_{i=0}^{m-1} f_i(v)\beta_i$$

der in der Basis B gegebenen Operanden v und δ zu bestimmen. Dann gilt

$$v \cdot \delta^q = \sum_{i=0}^{m-1} f_{i-1}(v^q)\beta_i \ .$$

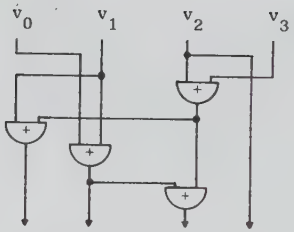
d.h. die Produkte $v \cdot \delta$ und $v \cdot \delta^q$ können mit derselben Hardware berechnet werden.

(2.20) Beispiel:

Sei $q = 2$, $m = 4$ und α Nullstelle des irreduziblen Polynoms $x^4 + x + 1 \in GF(2)[x]$. Dann ist $B = \langle \alpha^3, \alpha^6, \alpha^{12}, \alpha^9 \rangle$ eine Normalbasis von GF(16) über GF(2) und es gilt

$$v\alpha = (v_1+v_2+v_3 \ , v_0+v_1 \ , v_0+v_1+v_2+v_3 \ , v_2)$$
$$v\alpha^2 = (v_3 \ , v_0+v_2+v_3 \ , v_1+v_2 \ , v_0+v_1+v_2+v_3)$$
$$v\alpha^4 = (v_0+v_1+v_2+v_3 \ , v_0 \ , v_0+v_1+v_3 \ , v_2+v_3)$$
$$v\alpha^8 = (v_0+v_3 \ , v_0+v_1+v_2+v_3 \ , v_1 \ , v_0+v_1+v_2)$$

Abbildung (2.21) stellt die aus Beispiel (2.20) erhaltene parallele Implementierung der Multiplikation mit der Konstanten α dar.



(2.21) **Abbildung**: Multiplikation mit α

§3 Berechnung von multiplikativen Inversen

Zur Berechnung des multiplikativen Inversen eines von Null verschiedenen Elementes ω von $GF(q^m)$ werden im folgenden zwei serielle Verfahren vorgestellt. Liegt ω in polynomialer Basis vor, so kann ω^{-1} mit einer aus einer Modifikation des erweiterten Euklid'schen Algorithmus für Polynome (siehe z.B. /Li81/) abgeleiteten Schaltung bestimmt werden.

(3.1) Euklid'scher Algorithmus:

```
EUKLID () {
        f(x) = a(x);   t_f(x) = 0;   s_f(x) = 1;
        g(x) = b(x);   t_g(x) = 1;   s_g(x) = 0;
        do {  /*    f(x) = s_f(x)a(x) + t_f(x)b(x)        */
              /*    g(x) = s_g(x)a(x) + t_g(x)b(x)        */
              /*    ggT(a(x),b(x)) = ggT(f(x),g(x))       */
              exchange (f(x),g(x));
              exchange (t_f(x),t_g(x));
              exchange (s_f(x),s_g(x));
              q(x) = f(x) div g(x);
              f(x) = f(x) - q(x)g(x);
              t_f(x) = t_f(x) - q(x)t_g(x);
              s_f(x) = s_f(x) - q(x)s_g(x); }
        while ( f(x) != 0 ) }
/*    ggT(a(x),b(x)) = g(x) = s_g(x)a(x) + t_g(x)b(x)   */
```

(3.2) Beobachtung:

Sei $\omega \in GF(q^m)$ in der polynomialen Basis $B = <1,\alpha,\ldots,\alpha^{m-1}>$ gegeben und bezeichne M(x) das Minimalpolynom von α. Dann gilt $ggT(M(x),\omega(x)) = 1$ und Algorithmus (3.1) berechnet mit $a(x) = M(x)$ und $b(x) = \omega(x)$

$$ggT(M(x),\omega(x)) = 1 = s_g(x)M(x) + t_g(x)\omega(x)$$

und es gilt $t_g(x)\omega(x) \equiv 1 \bmod M(x)$.

Für die Bestimmung von Inversen erübrigt sich folglich die Berechnung der Polynome $s_f(x)$ und $s_g(x)$. Der folgende Algorithmus (der hier nur in einer Version für binäre Polynome angegeben wird) vermeidet außerdem die explizite Durchführung der Polynomdivision.

(3.3) Modifizierter Euklid'scher Algorithmus:

```
EUKLID2 () {
      f(x) = M(x);  t_f(x) = 0;  g(x) = b(x);  t_g(x) = 1;
      do {  /*    ggT(M(x),b(x)) = ggT(f(x),g(x))    */
            if ( deg(g(x)) > deg(f(x)) ) {
                  exchange (f(x),g(x));
                  exchange (t_f(x),t_g(x)); }
            d = deg(f(x)) - deg(g(x));
            f(x) = f(x) - x^d g(x);
            t_f(x) = t_f(x) - x^d t_g(x); }
      while ( f(x) != 0 ) }
   /*    ggT(M(x),b(x)) = g(x) = 1 ; t_g(x)b(x) ≡ 1 mod M(x) */
```

(3.4) **Bemerkung**:

Für binäre Polynome kann der modifizierte Euklid'sche Algorithmus einzig unter Verwendung von Schiebeoperationen und Additionen durchgeführt werden. Das Verfahren ist mit zu m proportionalem Aufwand implementierbar, wobei die Bestimmung eines Inversen höchstens 2·m Takte erfordert, da sich mit jeder Schiebeoperation die Zahl der für die Berechnung relevanten Koeffizienten von f(x) oder g(x) erniedrigt.

(3.5) **Beispiel**:

Sei q = 2, m = 4 und B = $<1,\alpha,\alpha^2,\alpha^3>$ polynomiale GF(2)-Basis von $GF(2^4)$ mit $\alpha^4 = \alpha+1$. Mit Algorithmus (3.3) kann $\alpha^{-4} = \alpha^3+\alpha^2+\alpha$ in 2·4 = 8 Takten bestimmt werden. Der Ablauf dieser Berechnung ist in der untenstehenden Tabelle angegeben.

i	f(x)	g(x)	$t_f(x)$	$t_g(x)$
0	11001	11000	00000	10000
1	11001	.1100	00000	10000
2	11001	..110	00000	10000
3	11001	...11	00000	10000
+	11010	...11	00010	10000
4	.1101	...11	00010	10000
+	.1110	...11	00110	10000
5	..111	...11	00110	10000
<+>	...11	..100	10000	01110
6	...11	...10	10000	01110
7	...11	1	10000	01110
+	...10	1	10111	01110
8	1	1	10111	01110
+	0	1	11001	01110

(3.6) Bemerkungen:

(a) Von E.R.Berlekamp wurde ein ebenfalls auf dem Euklid'schen Algorithmus basierendes Verfahren zur Berechnung von Inversen vorgeschlagen (cf. /Be68/).

(b) T.C.Bartee und D.I.Schneider bestimmen das Inverse eines in polynomialer Basis gegebenen $GF(q^m)$-Elementes durch Inversion der Begleitmatrix (cf. /Ba63/).

Eine weitere Möglichkeit zur Berechnung von multiplikativen Inversen ergibt sich aus:

(3.7) Beobachtung:

Für das multiplikative Inverse eines von Null verschiedenen Elementes ω von $GF(q^m)$ gilt (mit $Q = q^m$)

$$\omega^{-1} = \omega^{Q-2}.$$

(3.8) Beobachtung:

$$2^m-2 = \sum_{i=1}^{m-1} 2^i ,$$

d.h. ein Element $\omega \neq 0$ von $GF(2^m)$ kann mit dem Horner-Schema

$$(\ldots((\omega^2\omega)^2\omega)^2\ldots\omega)^2$$

durch m-2 Multiplikationen und (m-1)-maliges Quadrieren invertiert werden.

Ist der Operand ω in einer Normalbasis dargestellt, so erfordert das Quadrieren lediglich eine Permutation der Koeffizienten. Zusätzlich ermöglicht die Normalbasisdarstellung die Implementierung eines seriellen Invertierers.

(3.9) Beobachtung:

Sei $B = <\beta_0,\beta_1,\ldots\beta_{m-1}>$ Normalbasis von $GF(q^m)$ über $GF(q)$ mit $\beta_{i+1} = \beta_i^q$ und sei das multiplikative Inverse

$$w^{-1} = \sum_{i=0}^{m-1} u_i\beta_i$$

zu bestimmen. Sei ferner $u_i = f(w)$. Dann gilt

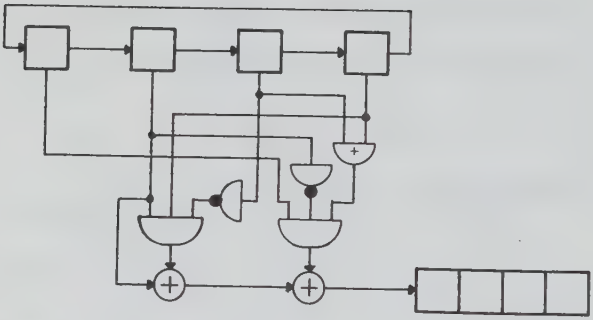
$$u_{i-1} = f(w^q).$$

(3.10) Beispiel:

Sei $q = 2$, $m = 4$ und α Nullstelle des irreduziblen Polynoms $x^4 + x + 1 \in GF(2)[x]$. Dann ist $B = <\alpha^3,\alpha^6,\alpha^{12},\alpha^9>$ eine Normalbasis von GF(16) über GF(2) und man erhält (nach mühsamer Berechnung von $w^{-1} = ((w^2w)^2w)^2$)

$$u_3 = w_1 + w_1\bar{w}_2w_3 + w_0\bar{w}_1(w_2+w_3).$$

Die aus Beispiel (3.10) resultierende Implementierung der Invertierung in GF(16) ist in Abbildung (3.11) dargestellt.



(3.11) Abbildung: Serieller Normalbasis-Invertierer

II. Walsh-Hadamard-Transformation und lineare Codes

Die Walsh-Hadamard-Transformation (WHT) zählt neben der Diskreten Fourier-Transformation zu den bekanntesten Vertretern orthogonaler Transformationen. Die Bedeutung der WHT in der digitalen Signalverarbeitung beruht nicht zuletzt auf der Tatsache, daß zu ihrer Implementierung nur Additionen und Subtraktionen erforderlich sind. Darüber hinaus sind schnelle Algorithmen zur Berechnung der WHT verfügbar (cf. /Ah75/).

Die bekannteste Anwendung der Walsh-Hadamard-Transformation in der Codierungstheorie ist das in der 'Green-Machine' realisierte Decodierverfahren für Reed-Muller-Codes (eingesetzt bei den Mariner-Projekten der NASA zwischen 1969 und 1977, cf. /Ma78/). In den folgenden Abschnitten wird gezeigt, daß die WHT auch bei der Bestimmung der Gewichtsverteilung linearer Codes (siehe §2) und bei der Konstruktion verschiedener Klassen "guter" Codes (§3) von Nutzen sein kann.

§1 Die Walsh-Hadamard-Transformation

(1.1) Definition:

Eine ***Hadamard-Matrix*** ist eine (r×r)-Matrix H_r mit Einträgen aus {1,-1} und der Eigenschaft

$$H_r \cdot H_r^t = r \cdot I_r.$$

(1.2) Bemerkungen (cf. /Be85/):

(a) Für eine (r×r)-Hadamard-Matrix gilt $r = 1$, $r = 2$ oder $r \equiv 0 \bmod 4$.

(b) Sind H_r und $H_{r'}$ Hadamard-Matrizen, so ist $H_r \otimes H_{r'}$ eine (rr'×rr')-Hadamard-Matrix.

(1.3) Beispiele:

Die Matrizen

$$H_1 = \begin{bmatrix} 1 \end{bmatrix} \quad \text{und} \quad H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

sind Hadamard-Matrizen. Mit Bemerkung (1.2b) können aus H_2 für alle natürlichen Zahlen i $(2^i \times 2^i)$-Hadamard-Matrizen (die sogenannten ***Sylvester-Matrizen***) konstruiert werden.

(1.4) **Definition**:

Eine Hadamard-Matrix $H_r = ((h_{i,j}))$ mit $h_{0,i} = h_{i,0} = 1$ für $0 \leq i < r$ heißt *normalisiert*.

Die aus einer normalisierten Hadamard-Matrix H_r durch Ersetzen von '1' durch '0' und '-1' durch '1' erhaltene Matrix B_r heißt *binäre Hadamard-Matrix*.

Binäre Hadamard-Matrizen sind offenbar nicht invertierbar. Wie man leicht nachrechnet, gilt jedoch:

(1.5) **Beobachtung**:

Sei H_r eine $(2^s \times 2^s)$-Sylvester-Matrix und seien H_r^* und B_r^* die aus H_r bzw. B_r durch Streichen der ersten Zeile und der ersten Spalte erhaltenen $((2^s-1)\times(2^s-1))$-Matrizen. Dann gilt

$$(B_r^*)^{-1} = \frac{H_r^*}{-2^{s-1}}.$$

(1.6) **Bemerkungen**:

(a) Da die Zeilen einer Hadamard-Matrix H_r paarweise orthogonal sind, beträgt der Hamming-Abstand zweier verschiedener Zeilen einer binären Hadamard-Matrix B_r $r/2$.

(b) Durch Weglassen der ersten Spalte einer binären Hadamard-Matrix B_r und Interpretation der Zeilen der Restmatrix als Codeworte eines binären Codes C erhält man einen Code der Mächtigkeit r, der Länge r-1 und der Minimaldistanz r/2. Im allgemeinen ist C nicht linear.

(1.7) **Definition**:

Sei H_r eine (r×r)-Hadamard-Matrix. Die **Walsh-Hadamard-Transformierte** eines Vektors $\underline{c}=(c_0,c_1,\ldots,c_{r-1})$ (mit $c_i \in \mathbb{R}$) ist der Vektor

$$\underline{C} = \underline{c} \cdot H_r .$$

(1.8) **Bemerkung**:

Die Walsh-Hadamard-Transformation kann für $r = 2^s$ - nur dieser Fall ist im weiteren von Interesse - unter Ausnutzung von Bemerkung (1.2b) in $O(s \cdot 2^s)$ Schritten berechnet werden (cf. /Ru69/, /Ah75/).

§2 WHT und die Gewichtsverteilung linearer Codes

Die Gewichtsverteilung eines binären linearen Codes C der Dimension k kann durch die Multiplikation eines Vektors der Länge 2^k über den natürlichen Zahlen, in dem die Spalten einer Generatormatrix von C codiert sind, mit einer $(2^k \times 2^k)$-Matrix M_k, die nur von k abhängt, berechnet werden (cf. /Gr60/, /Ma60/).

(2.1) **Beobachtung**:

Sei C ein (n,k)-Code über GF(2) mit Generatormatrix $G = ((g_{i,j}))$, wobei $g_{i,j} \in \{0,1\}$. Sei ferner $a = 2^k$ und der Vektor $\underline{N} = (N_0, N_1, \ldots, N_{a-1})$ definiert durch

$$N_j = |\{ e \in [0:n-1] \mid \sum_{i=0}^{k-1} g_{i,e} \cdot 2^i = j \}| \qquad (0 \leq j < a)$$

d.h. N_j gibt die Anzahl der Spalten von G an, die der Binärdarstellung der Zahl j (im folgenden mit $(j)_2$ bezeichnet) entsprechen. Sei schließlich durch

$$m_{i,j} = wgt((i)_2 \text{ AND } (j)_2) \bmod 2 \qquad (0 \leq i,j < a)$$

die (a×a)-Matrix $M_k = ((m_{i,j}))$ gegeben und sei

$$\underline{W} = (W_0, W_1, \ldots, W_{a-1}) = \underline{N} \cdot M_k .$$

Dann gilt:

$$W_i = wgt((i)_2 \cdot G) \qquad (0 \leq i < a).$$

Beweis: Zu W_i trägt eine Spalte $\underline{s}$ der Generatormatrix G genau dann bei, wenn gilt

$$\text{wgt}((i)_2 \text{ AND } \underline{s}) \bmod 2 = \left(\sum_{j=0}^{k-1} i_j \cdot s_j\right) \bmod 2 = 1.$$

□

(2.2) **Beobachtung:**

M_k ist eine binäre Hadamard-Matrix.

Beweis: Induktion über k.

□

(2.3) **Folgerung:**

Besitzt die Generatormatrix G eines (n,k)-Codes über GF(2) keine Null-Spalte, so gilt mit $a = 2^k$:

$$\sum_{i=0}^{a-1} W_i = n \cdot (a/2) .$$

(2.4) Bemerkungen:

(a) Mit elementaren Methoden der Kombinatorik kann ein Folgerung (2.3) entsprechendes Resultat für (n,k)-Codes C über GF(q) gezeigt werden (cf. /Be82b/):

$$\sum_{\underline{c}\in C} wgt(\underline{c}) = n\cdot(q^{k}-q^{k-1}) \ .$$

(b) Die Beschreibung eines linearen Codes durch den Vektor $\underline{N}$ der Anzahlen der in der Generatormatrix auftretenden Spalten jedes Typs wird **modulare Repräsentation** genannt (cf. /Pe72/).

(c) Die Berechnung der Gewichte aller Codeworte von C kann mit schneller "binärer" Hadamard-Transformation in O(a·log(a)) Schritten durchgeführt werden.

§3 WHT und die Konstruktion linearer Codes

Da das Minimalgewicht eines Codes das mittlere Gewicht der vom Nullvektor verschiedenen Codeworte nicht übersteigen kann, folgt für einen (n,k)-Code über GF(2) mit (2.3) sofort:

(3.1) **Beobachtung**:

$$n \geq ((2^k-1)\cdot d) \,/\, 2^{k-1}.$$

(3.2) **Bemerkungen**:

(a) Wegen $(2^k-1, 2^{k-1}) = 1$ kann in (3.1) die Gleichheit höchstens für $d \equiv 0 \bmod 2^{k-1}$ erreicht werden.

(b) Die durch Beobachtung (3.1) gegebene untere Schranke für die Länge eines binären linearen Codes mit Dimension k und Minimaldistanz d entspricht der Plotkin-Schranke (siehe (I.2.8) und /Be82b/).

(3.3) **Beobachtung**:

Sei $n = (2^k-1)\cdot s$ und $\underline{N} = (0,s,s,\ldots,s)$. Dann ergibt sich

$$\underline{W} = (0, s\cdot 2^{k-1}, s\cdot 2^{k-1}, \ldots, s\cdot 2^{k-1}).$$

Die für s = 1 resultierenden Codes sind die zu den binären Hamming-Codes dualen Simplex-Codes (cf. /Ma78/).

(3.4) Satz:

Sei C ein (n,k)-Code über GF(2) mit Minimaldistanz d und sei e die natürliche Zahl mit $2^{e-1} < d \leq 2^e$. Dann gilt

(a) $N(k+1,\min(2d,n),2) \leq 2n$ und

(b) für $s \leq e+1$:

$$N(k+s,d,2) \leq n+(2^s-1)((d+2^{s-1}-1) \text{ div } 2^{s-1}) < n+4d.$$

Beweis: Sei $\underline{N}$ die modulare Repräsentation von C und $\underline{W} = \underline{N} \cdot M_k$.

(a) $\underline{N}' = (\underline{N},\underline{N})$ ist die modulare Repräsentation eines $(2n,k+1)$-Codes C' mit $\underline{W}' = \underline{N}' \cdot M_{k+1} = (2 \cdot \underline{W},n,n,\ldots,n)$ und Minimaldistanz $d' = \min(2d,n)$.

(b) Sei $r = (d+2^{s-1}-1) \text{ div } 2^{s-1}$ und $\underline{R}$ modulare Repräsentation eines beliebigen (r,k)-Codes. Dann ist $\underline{N}' = (\underline{N},\underline{R},\underline{R},\ldots,\underline{R})$ die modulare Repräsentation eines $(n+r \cdot 2^s-1,k+s)$-Codes C' mit Minimaldistanz $d' \geq d$. Weiterhin gilt

$$(2^s-1)((d+2^{s-1}-1) \text{ div } 2^{s-1}) \leq (2^s-1)(2^{e-s+1}) = 2^{e+1}-2^{e-s+1} < 4d.$$

□

(3.5) Beispiel:

Sei C der binäre (6,3)-Code mit Generatormatrix

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

C besitzt die Minimaldistanz $d = 3$ und die modulare Repräsentation $\underline{N} = (0,1,1,1,0,1,1,1)$. Die iterierte Anwendung von (3.4a) liefert eine Folge von $(3\cdot 2^{k-2},k)$-Codes mit der Minimaldistanz $d = n/2$ und dem Gewichtsvektor $\underline{W} = (0,2^{k-1},2^{k-1},2^{k-1},d,d,\ldots,d)$. Diese Codes sind optimal bezüglich der Schranke (3.1).

III. Diskrete Fourier-Transformation und zyklische Codes

Die Fourier-Transformation ist zweifellos die prominenteste der heute benutzten orthogonalen Transformationen. Die enorme Bedeutung der Diskreten Fourier-Transformation (DFT) im Bereich der digitalen Signalverarbeitung beruht zum einen auf ihren vielfältig nutzbaren Eigenschaften, zum anderen ebenso entscheidend auf der Verfügbarkeit schneller Berechnungsverfahren - ein Gebiet, auf das an dieser Stelle nicht näher eingegangen werden kann (hierzu siehe z.B. /Nu81/ und /Be84b/).

Fourier-Transformation über endlichen Körpern ist inzwischen in der Codierungstheorie (nicht selten unter der Bezeichnung *Mattson-Solomon-Polynom*) als hilfreiches Instrument für die Beschreibung und die Analyse zyklischer Codes anerkannt. Ihre Bedeutung für die Implementierung von Codes ist dagegen gering geblieben; ihre Eliminierung aus den Standardverfahren zur Decodierung von BCH-Codes das Ziel neuerer Arbeiten (cf. /Bl82/, /Bl84/). Beides wird u.a. mit dem hohen Aufwand zur Berechnung der Fourier-Transformation begründet; dies obwohl hierfür die oben erwähnten "schnellen" Algorithmen zur Verfügung stehen. Das in §3 entwickelte Verfahren zur Berechnung der inversen DFT (ein neuer Zugang zur inversen ADFT; vgl. Kap.IV) ist aufgrund seiner Effizienz geeignet, die Attraktivität DFT-basierter Codier- und Decodierverfahren zu steigern.

§1 Die Diskrete Fourier-Transformation

(1.1) **Definition**:

Die ***Diskrete Fourier-Transformierte*** (DFT) eines Signalvektors $\underline{c} = (c_0, c_1, \ldots, c_{n-1})$ ist der Spektralvektor $\underline{C} = (C_0, C_1, \ldots, C_{n-1})$, der durch die Gleichung $\underline{C} = D_n \underline{c}$ bestimmt ist, wobei D_n die (n×n)-***Fouriermatrix*** $((\alpha^{ik}))_{i,k=0,1,\ldots,n-1}$ bezeichnet, deren Einträge aus den Potenzen einer primitiven n-ten Einheitswurzel α bestehen.

Fourier-Transformation entspricht der Auswertung eines (Signal-) Polynoms an den n-ten Einheitswurzeln, die inverse Fourier-Transformation der Interpolation eines (Signal-) Polynoms durch n gegebene (Spektral-) Koeffizienten. Eine Fourier-Transformation der Länge n in einem endlichen Körper GF(q) existiert genau dann, wenn n und q relativ prim sind.

Die ***polynomiale Interpretation*** der DFT legt ihre serielle Berechnung mit Hilfe linearer Rückkopplungs-Schieberegistern unmittelbar nahe:

(1.2) **Beobachtung**:

Für die zu einem Signalvektor $\underline{c}$ bzw. dem assoziierten Signalpolynom c(x) gehörenden Spektralkoeffizienten gilt:

$$C_j = c(\alpha^j) = c(x) \bmod (x - \alpha^j) \qquad (\ 0 \leq j < n\ ).$$

Die in diesem Rahmen benötigten weiteren Eigenschaften der DFT werden im folgenden ohne Beweis angegeben. Für eine ausführlichere Darstellung sei auf /Po71/, /Nu81/, /Bl83/ und /Be84/ verwiesen.

(1.3) Inversionseigenschaft:

Die Inverse zur Fouriermatrix $D_n = ((\alpha^{ik}))_{i,k=0,1,\ldots,n-1}$ ist gegeben durch

$$D_n^{-1} = (1/n)\cdot((\alpha^{-ik}))_{i,k=0,1,\ldots,n-1}.$$

(1.4) Bemerkung:

Die polynomiale Interpretation der inversen DFT ergibt sich unmittelbar aus der Inversionseigenschaft:

$$c_j = (1/n)\cdot C(\alpha^{-j}) = (1/n)\cdot C(x) \bmod (x - \alpha^{-j}).$$

(1.5) Folgerung:

(a) Ein Polynom p(x) aus dem Signalbereich besitzt α^i als Nullstelle genau dann, wenn sein i-ter Spektralkoeffizient Null ist.

(b) Ein Polynom P(x) aus dem Spektralbereich besitzt α^{-i} als Nullstelle genau dann, wenn sein i-ter Signalkoeffizient Null ist.

(1.6) **Phaseneigenschaft:**

Für die Spektralkoeffizienten des zu einem Signalvektor $\underline{c} = (c_0, c_1, \ldots, c_{n-1})$ zyklisch verschobenen Signals $\underline{c}' = (c_d, c_{d+1}, \ldots, c_{d+n-1})$ gilt:

$$C_j' = C_j \, \alpha^{-jd}.$$

(1.7) **Skalierungseigenschaft:**

Für die Spektralkoeffizienten des gegenüber einem Signalvektor $\underline{c} = (c_0, c_1, \ldots, c_{n-1})$ durch einen Skalierungsfaktor $s > 0$ mit $(n,s) = 1$ permutierten Signals $\underline{c}' = (c_0, c_s, \ldots, c_{(n-1)s})$ gilt:

$$C_j' = C_{j/s}.$$

(1.8) **Faltungseigenschaft:**

Das Spektrum $\underline{U}$ der zyklischen Faltung zweier Signalvektoren $\underline{c}$ und $\underline{h}$ kann durch das koeffizientenweise Produkt ihrer Transformierten berechnet werden:

$$U_j = C_j \cdot H_j.$$

(1.9) **Bemerkung:**

Mit Hilfe der Inversionseigenschaft der DFT können nach Vertauschen von "Signal" und "Spektrum" duale Versionen der Eigenschaften (1.6) bis (1.8) formuliert werden.

§2 DFT und die Beschreibung zyklischer Codes

Ein (n,k)-Code C über GF(q) kann durch die Spektren seiner Codewörter beschrieben werden, falls (n,q) = 1 gilt und somit eine DFT zu den Parametern n und q existiert. Ein zyklischer Code ist durch die Menge seiner Nullstellen definiert. Dies führt sofort zu:

(2.1) Beobachtung:

Sei C ein zyklischer (n,k)-Code über GF(q), (n,q) = 1, α primitive n-te Einheitswurzel und Z die Menge der Nullstellen von C. Dann gilt für das DFT-Spektrum $\underline{C}$ eines Codewortes $\underline{c}$:

$$\alpha^i \in Z \implies C_i = 0 \qquad (0 \leq i < n).$$

Jedes Codewort c(x) eines zyklischen (n,k)-Codes C ist Vielfaches des Generatorpolynoms g(x); sei etwa

$$c(x) = f(x) \cdot g(x).$$

Dieses Produkt kann als zyklische Faltung im Signalbereich aufgefaßt und unter Ausnutzung der Faltungseigenschaft der DFT durch koeffizientenweise Multiplikation der Transformierten F(x) und G(x) und Rücktransformation des Resultats aus dem Spektralbereich realisiert werden. Da f(x) beliebig gewählt werden kann, folgt:

(2.2) Beobachtung:

Jedes Spektrum $\underline{C}$ mit

$$C_j = F_j \cdot G_j \qquad (0 \leq j < n),$$

dessen zugehöriges Signal $\underline{c}$ im Vektorraum $GF(q)^n$ liegt, ist ein *Codewort im Frequenzbereich*. Das Spektrum $\underline{G}$ des Generatorpolynoms $g(x)$ legt die für jedes Codewort verschwindenden Spektralkoeffizienten, die *Kontrollfrequenzen*, fest.

Die Freiheit bei der Wahl des informationstragenden Anteils $\underline{F}$ ist durch Nebenbedingungen (*'conjugacy constraints'*) eingeschränkt.

(2.3) Satz:

Für die inverse Fourier-Transformierte $\underline{v}$ eines Vektors $\underline{V} \in GF(q^m)^n$ gilt $\underline{v} \in GF(q)^n$ genau dann, wenn für $0 \leq j < n$

$$v_j^q = v_{jq}.$$

Beweis: Sei α primitive n-te Einheitswurzel mit

$$V_j = \sum_{i=0}^{n-1} \alpha^{ij} v_i$$

Dann gilt für $\underline{v} \in GF(q)^n$

$$(*) \quad v_j^q = \sum_{i=0}^{n-1} \alpha^{ijq} v_i^q = \sum_{i=0}^{n-1} \alpha^{ijq} v_i = V_{jq} \qquad (0 \leq j < n).$$

Wegen $(n,q) = 1$ ist $i \dashrightarrow i \cdot q$ Permutation von $[0:n-1]$, so daß andererseits aus (*) $v_i = v_i^q$ für $0 \leq i < n$ und damit $\underline{v} \in GF(q)^n$ folgt.

□

Satz (2.3) zeigt die Redundanz im DFT-Spektrum für den (nicht unüblichen) Fall $m > 1$ auf. Das Spektrum eines Signalvektors aus $GF(q)^n$ ist durch je einen Spektralkoeffizienten als Repräsentant jeder Kreisteilungsklasse bereits völlig bestimmt.

(2.4) Bemerkungen:

(a) Ein BCH-Code zur Entwurfsdistanz d' besitzt d'-1 konsekutive Kontrollfrequenzen.

(b) Für Reed-Solomon-Codes gilt $m = 1$, d.h. es gelten keine Einschränkungen für die Wahl von $\underline{F}$.

Zur Beschreibung verallgemeinerter Reed-Solomon-Codes mittels DFT erweist es sich für den Fall $n < q-1$ als günstig, den Code als zur Länge q-1 trivial ergänzt zu betrachten:

(2.5) Beobachtung:

Sei C der durch $\alpha = (\alpha_0,\alpha_1,\ldots,\alpha_{s-1})$ mit $\alpha_i \in GF(q)^*$, $\alpha_i \neq \alpha_j$ für $i \neq j$ und $\nu = (\nu_0,\nu_1,\ldots,\nu_{s-1})$ mit $\nu_i \in GF(q)^*$ definierte (s,k)-GRS-Code über GF(q). Sei β primitive (q-1)-te Einheitswurzel und $\underline{v} = (v_0,v_1,\ldots,v_{q-2})$ gegeben durch

$$v_j = \begin{cases} \nu_i/(q-1) & \text{falls für i gilt: } \beta^{-j} = \alpha_i \\ \\ 0 & \text{sonst.} \end{cases}$$

Dann ist für $f(x) \in GF(q)[x]$ mit $\deg(f(x)) < k$ das Spektrum des Vektors

$$\underline{c} = (v_0 f(1), v_1 f(\beta^{q-2}), v_2 f(\beta^{q-3}), \ldots, v_{q-2} f(\beta))$$

die Faltung der Spektren von $\underline{v}$ und $\underline{f}$.

(2.6) Bemerkung:

Bei Alternant-Codes gelten für die Koeffizienten dieser Faltung die Nebenbedingungen aus Satz (2.3).

In Tabelle (2.7) sind die betrachteten Codeklassen und ihre Beschreibung mittels DFT zusammengefaßt.

Code	Länge	Kontroll-frequenzen	davon konsekutiv	Neben-bedingungen
zyklisch	(n,q)=1	n-k	-	ja
BCH	(n,q)=1	n-k	d'-1	ja
RS	n|(q-1)	d-1	d-1	nein
GRS	$n \leq q-1$	n-k	n-k	nein
Alternant	$n \leq q^m-1$	n-k	n-k	ja

(2.7) Tabelle: Codeklassen und ihre Beschreibung mittels DFT

Abschließend soll als Beispiel für die Anwendung der vorgestellten Charakterisierung zyklischer Codes durch die Diskrete Fourier-Transformation die Gültigkeit der als BCH-Schranke (vgl. Bemerkung (I.3.22)) bekannten unteren Abschätzung für die Minimaldistanz zyklischer Codes mit Mitteln der DFT bewiesen werden.

(2.8) Satz:

Ein Vektor $\underline{v}$ in $GF(q)^n$ mit $wgt(\underline{v}) \leq d'$, dessen Spektrum d' konsekutive Nullstellen besitzt, ist der Nullvektor.

Beweis: Sei $r = wgt(\underline{v})$ und $R = Tr(\underline{v})$ die Menge der von Null verschiedenen Koeffizienten von $\underline{v}$. Das zu dem durch

$$F(x) = \prod_{i \in R} (1-\alpha^i x) = F_r x^r + F_{r-1} x^{r-1} + \ldots + F_0$$

definierten Spektralpolynom gehörende Signalpolynom $f(x)$ besitzt nach Folgerung (1.5b) die Eigenschaft

$$v_i \neq 0 \implies f_i = 0 \qquad (0 \leq i < n),$$

d.h.

$$v_i \cdot f_i = 0 \qquad (0 \leq i < n),$$

so daß für die Faltung der Spektren

$$V(x) \cdot F(x) = 0$$

gelten muß. Da $F_i = 0$ für $i > d'$ und $F_0 = 1$, sowie d' konsekutive Koeffizienten von $V(x)$ Null sind, folgt wegen

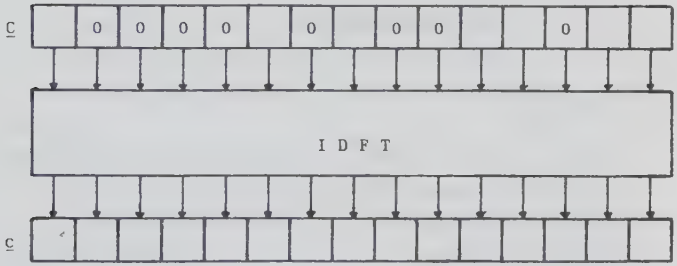
$$V_j = - \sum_{k=1}^{d'} F_k V_{j-k}$$

$V(x) = 0$ und damit $\underline{v} = 0$.

□

§3 DFT und die Implementierung zyklischer Codes

Zur Codierung eines zyklischen Codes mittels DFT (cf. /Bl83/) wendet man auf ein Codewort im Spektralbereich die inverse DFT an und erzeugt somit ein (nicht systematisch codiertes) Codewort im Signalbereich.



(3.1) Abbildung: DFT-Codierer für den binären (15,7)-BCH-Code

(3.2) Bemerkung:

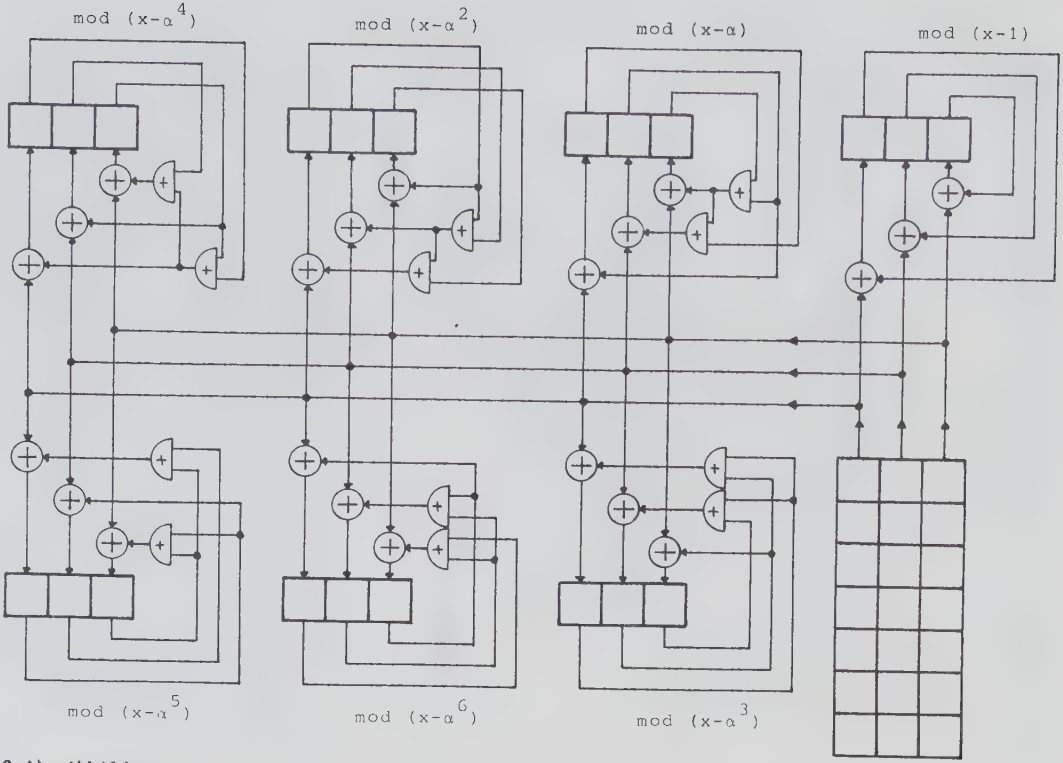
Sei C ein zyklischer (n,k)-Code über GF(q). Ein Codewort im Spektralbereich erhält man durch Nullsetzen der n-k Kontrollfrequenzen von C und Verteilen der k GF(q)-Informationssymbole auf die verbleibenden Koeffizienten eines Vektors $\underline{C} \in GF(q^m)^n$ unter Beachtung der durch Satz (2.3) gegebenen Nebenbedingungen.

Die Berechnung der IDFT erfordert $GF(q^m)$-Arithmetik, deren Implementierung von der gewählten GF(q)-Basis von $GF(q^m)$ abhängt (siehe Kapitel I). Die Wahl einer Normalbasisdarstellung erweist sich als vorteilhaft:

- die Konjugierten eines Elementes β von $GF(q^m)$ erhält man durch Permutation der Koeffizienten von β;
- die Multiplikation mit $\alpha \in GF(q^m)$ kann mit derselben Hardware wie die Multiplikation mit α^q durchgeführt werden (siehe Beobachtung (I.2.19)).

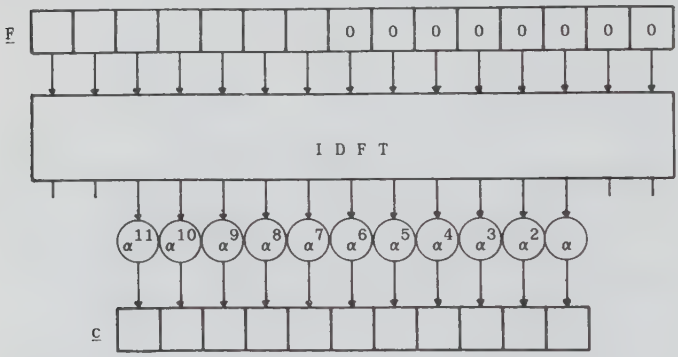
(3.3) Beispiel:

Sei $q = 2$, $n = 7$ und α eine primitive 7-te Einheitswurzel mit Minimalpolynom x^3+x+1. Der durch die Nullstellen $\{\alpha,\alpha^2\}$ definierte BCH-Code C besitzt die Dimension $k = 4$ und die Minimaldistanz $d = 3$ (C ist ein (7,4)-Hamming-Code /Ma78/). C_0, C_3, C_5 und C_6 sind die von Null verschiedenen Spektralkoeffizienten eines Codewortes $\underline{c} \in C$. Wegen Satz (2.3) ist C_0 Element von GF(2) (1 Bit Information) und der ($GF(2^3)$-) Wert von (z.B.) C_3 bestimmt die Werte der Koeffizienten C_5 und C_6 (3 Bit Information). $B = <\alpha^3,\alpha^6,\alpha^5>$ ist Normalbasis von $GF(2^3)$ über GF(2). In Abbildung (3.4) ist die auf dieser Basis basierende Implementierung eines seriellen DFT-Codierers für den (7,4)-Hamming-Code dargestellt.



(3.4) **Abbildung**: Serieller DFT-Codierer für den (7,4)-Hamming-Code

Die Codierung verallgemeinerter Reed-Solomon-Codes mittels DFT zeigt Abbildung (3.5) am Beispiel des (11,7)-GRS-Codes über GF(16) mit $\alpha = \nu = (\alpha^3, \alpha^4, \ldots, \alpha^{13})$.



(3.5) **Abbildung**: DFT-Codierer für einen (11,7)-GRS-Code

Die Codierung zyklischer Codes mit Hilfe der DFT (vgl. /Bl83/) erscheint im Vergleich zu den bekannten Standard-Algorithmen (siehe z.B. /Pe72/, /Ma78/) zumindest für $m > 1$ als unnötig aufwendig und für die Praxis wenig geeignet. In diesem Fall stimmen (wie auch bei vielen weiteren Anwendungen der DFT, cf. /Be83b/) Signal- und Spektralbereich nicht überein; vielmehr ist der Signalbereich ein Unterkörper des Spektralbereichs. Im folgenden wird diese Situation zur Herleitung eines Verfahrens zur Berechnung der inversen DFT

ausgenutzt, das sich durch deutlich geringere Komplexität auszeichnet.

Sei $F(x) \in GF(q^m)[x]$ DFT-Spektrum eines Signalpolynoms $f(x) \in GF(q)[x]$, sei $B = <\beta_0,\beta_1,\ldots,\beta_{m-1}>$ Normalbasis von $GF(q^m)$ über $GF(q)$ mit $\beta_{i+1} = \beta_i^q$ und gelte

$$F(x) = \sum_{i=0}^{n-1}{}' F_i x^i \qquad \text{wobei} \qquad F_i = \sum_{j=0}^{m-1} e_{ij}\beta_j .$$

Sei ferner α primitive n-te Einheitswurzel in $GF(q^m)$ und sei

$$e_j(x) = \sum_{i=0}^{n-1} e_{ij}x^i$$

Mit Satz (2.3) erhält man sofort:

(3.6) <u>Beobachtung</u>:

$$e_{j+1}(x) = e_j(x)^q .$$

Der folgende Satz führt zu einem effizienten Verfahren zur Bestimmung der Koeffizienten von f(x).

(3.7) Satz:

Sei $D = \langle \delta_0, \delta_1, \ldots, \delta_{m-1} \rangle$ zur Basis B spur-duale GF(q)-Basis in $GF(q^m)$ und sei

$$w = e_0(\alpha) = \sum_{j=0}^{m-1} w_j \delta_j .$$

Dann gilt $F(\alpha) = w_0$.

Beweis: Wegen

$$F(x) = \sum_{j=0}^{m-1} e_j(x)\beta_j$$

gilt

$$F(\alpha) = \sum_{j=0}^{m-1} e_j(\alpha)\beta_j = tr(e_0(\alpha)\cdot\beta_0).$$

und mit Lemma (I.1.15) folgt die Behauptung.

□

(3.8) Folgerung:

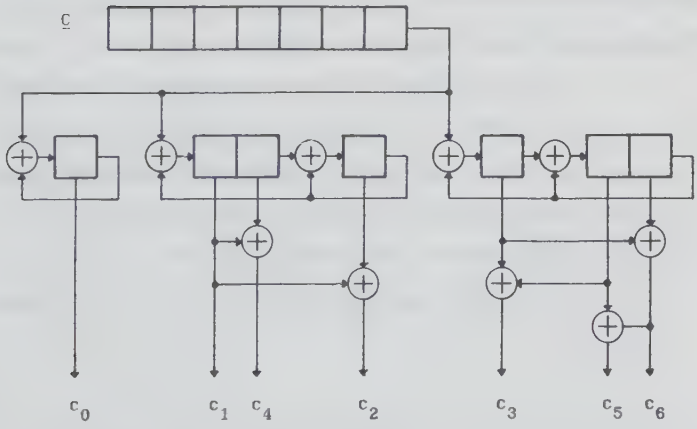
Für $0 \leq j < m$ und $i = q^j$ gilt $F(\alpha^i) = w_j$, d.h. die Koeffizienten des Signalpolynoms f(x) können durch Reduktion des Polynoms $e_0(x)$ modulo der Minimalpolynome der n-ten Einheitswurzeln und Transformation der erhaltenen Reste in die Basis D berechnet werden.

(3.9) Bemerkung:

Dieses Verfahren zur Berechnung der inversen DFT entspricht der seriell implementierten inversen Algebraischen DFT (siehe Kapitel IV).

(3.10) Beispiel:

Sei $q = 2$, $n = 7$ und α primitive 7-te Einheitswurzel mit Minimalpolynom x^3+x+1. Dann ist $B = \langle\alpha^3,\alpha^6,\alpha^5\rangle$ eine selbstduale Normalbasis von $GF(2^3)$ über GF(2) und der durch die Nullstellen $\{\alpha,\alpha^2\}$ definierte (7,4)-Hamming-Code (siehe Beispiel (3.3)) kann mit der in Abbildung (3.11) dargestellten Schaltung codiert werden.



(3.11) Abbildung: Serieller ADFT-Codierer für den (7,4)-Hamming-Code

(3.12) Bemerkungen:

(a) Von dem in Folgerung (3.8) beschriebenen Verfahren abweichende Implementierungen der inversen ADFT führen zu weiteren Möglichkeiten der Codierung zyklischer Codes mittels DFT (siehe Kapitel IV).

(b) Die Codierung eines zyklischen Codes C mittels DFT ist von der Dimension und den Kontrollfrequenzen von C unabhängig: *Ein* DFT-Prozessor kann zur Codierung *aller* zyklischen (und einer Anzahl linearer) Codes fester Länge über einem Körper GF(q) und allen Unterkörpern eingesetzt werden.

Im Hinblick auf die Decodierung zyklischer Codes spielt die DFT in mehrfacher Hinsicht eine Rolle (siehe z.B. /Bl83/). Hier sollen die folgenden Aspekte näher betrachtet werden:

- die algebraischen Standard-Algorithmen zur Decodierung von BCH-Codes (die prinzipiell auch zur Decodierung von Alternant- und Goppa-Codes einsetzbar sind; siehe z.B. /Be68/, /Bl83/) können als **Decodierung im Frequenzbereich** aufgefaßt und mit Hilfe von Fourier-Transformationen implementiert werden;

- alternativ können mittels DFT die genannten Decodierverfahren in den Zeitbereich transformiert werden (cf. /Bl83/, /Bl84/). Die Implementierung der *Decodierung im Zeitbereich* erfolgt dann ohne Fourier-Transformationen.

Im weiteren bezeichne C einen BCH-Code der Länge n über GF(q) mit den konsekutiven Nullstellen $\alpha,\alpha^2,\ldots,\alpha^{2t}$ (für eine primitive n-te Einheitswurzel α). Die Entwurfsdistanz d' von C ist damit 2t+1 und C ist zumindest t-fehler-korrigierend. Nahezu alle algebraischen Decodier-Algorithmen für BCH-Codes können (wie auch die im folgenden skizzierten) selbst bei größerer tatsächlicher Minimaldistanz von C nur bis zur Entwurfsdistanz d' korrigieren. Die bekannten Decodierverfahren, die eine Korrektur über die Entwurfsdistanz hinaus erlauben (siehe /Ha72b/, /Ha74/), sind aufgrund ihrer hohen Komplexität kaum von praktischer Bedeutung (cf. /Ma78/).

Das algebraische Standard-Decodierverfahren für BCH-Codes wurde von W.W.Peterson /Pe60/, R.T.Chien und E.R.Berlekamp /Be68/ entwickelt, arbeitet im Frequenzbereich und kann in drei Schritte zerlegt werden:

- Berechnung der Syndrome,
- Bestimmung des Fehlerstellenpolynoms L(x) und
- Ermittlung des (wahrscheinlichsten) Fehlervektors aus L(x).

Die Abschnitte 1 und 3 dieses Decodierverfahrens beinhalten Fourier-Transformationen.

(3.13) **Beobachtung:**

Sei $\underline{c}$ das gesendete Codewort, $\underline{y}$ der empfangene Vektor und bezeichne $\underline{e}$ den bei der Übertragung aufgetretenen Fehler, d.h. es gelte $\underline{y} = \underline{c}+\underline{e}$. Dann können durch Fourier-Transformation von $\underline{y}$ die 2t Syndrome

$$S_j = y(\alpha^j) = e(\alpha^j) = E_j \qquad (1 \leq j \leq 2t).$$

berechnet werden.

Mit den Syndromen sind 2t konsekutive Werte des Spektrums $\underline{E}$ von $\underline{e}$ bekannt. Gilt $S_j = 0$ für $1 \leq j \leq 2t$, so wird $\underline{e} = \underline{0}$ angenommen und die weiteren Schritte erübrigen sich. Im anderen Fall sind vom Decodierer die restlichen n-2t Koeffizienten von $\underline{E}$ zu bestimmen.

(3.14) Bemerkung:

Das DFT-Spektrum $F(x) \in GF(q^m)[x]$ eines Signalpolynoms $f(x) \in GF(q)[x]$ kann mit dem Verfahren ADFT berechnet werden, das im wesentlichen dem in Folgerung (3.8) beschriebenen Algorithmus zur Berechnung der IDFT entspricht (siehe Kapitel IV).

(3.15) Beobachtung:

Sei $v = wgt(\underline{e})$ die (dem Decodierer unbekannte) Anzahl der aufgetretenen Fehler und $V = Tr(\underline{e})$ die Menge der Fehlerstellen. Dann heißt

$$L(x) = \prod_{i \in V} (1-\alpha^i x),$$

das *Fehlerstellenpolynom*; die Fehlerstellen sind durch die Nullstellen von L(x) bestimmt. Sei l(x) das Signalpolynom zu L(x). Mit Folgerung (1.5b) beobachtet man

$$e_i \neq 0 \implies l_i = 0 \qquad (0 \leq i < n)$$

und damit

$$e_i \cdot l_i = 0 \qquad (0 \leq i < n)$$

so daß für die zugehörigen Spektren gilt

$$E(x) \cdot L(x) = 0.$$

Da $L_i = 0$ für $i > v$ und $L_0 = 1$ gilt, sowie 2t konsekutive Koeffizienten von E(x) bekannt sind, können v Gleichungen für die v unbekannten Koeffizienten von L(x) aufgestellt und gelöst werden.

Das obige Gleichungssystem ist i.a. nicht eindeutig lösbar. Im Sinne einer 'maximum-likelihood'-Decodierung ist eine Lösung mit minimalem v zu bestimmen. Das bekannteste Verfahren hierzu ist der **Berlekamp-Massey-Algorithmus** (cf. /Be68/, /Ma69/), mit dem ein Fehlerstellenpolynom minimalen Grades in 2t Iterationen berechnet werden kann.

(3.16) Beobachtung:

Aus v Koeffizienten von L(x) und v der bekannten 2t Koeffizienten von $\underline{E}$ können mittels der Beziehung

$$E_j = - \sum_{i=1}^{t} L_i \cdot E_{j-i}$$

die unbekannten Koeffizienten von $\underline{E}$ durch rekursive Ergänzung in n-2t Iterationen ermittelt werden (cf. /Bl83/).

(3.17) Bemerkung:

Berechnet man neben dem Fehlerstellenpolynom L(X) auch dessen formale Ableitung L'(x) und das **Fehlerwertepolynom**

$$W(x) = L(x) \cdot S(x) \bmod x^{2t}.$$

so können die n-2t Iterationen zur rekursiven Ergänzung des Fehlervektors vermieden werden, da dann die unter der Bezeichnung **Forney-Algorithmus** (cf. /Fo65/) bekannte Beziehung

$$l_i = 0 \quad ==> \quad e_i = \alpha^i \cdot (-W(\alpha^{-i})/L'(\alpha^{-i}))$$

zur direkten Bestimmung der Fehlerwerte ausgenutzt werden kann.

Der um die in Bemerkung (3.17) angegebenen Berechnungen erweiterte Berlekamp-Massey-Algorithmus wird als *Berlekamp-Algorithmus* bezeichnet (cf. /Be68/).

Die beiden skizzierten algebraischen Verfahren zur Decodierung von BCH-Codes arbeiten im Frequenzbereich; ihnen geht eine Fourier-Transformation des empfangenen Vektors (die Syndromberechnung) voraus und an ihrem Ende ist eine inverse DFT erforderlich. Ziel der von R.E.Blahut (cf. /BL83/ und /Bl84/) vorgeschlagenen Decodierung im Zeitbereich ist die Eliminierung dieser Fourier-Transformationen. Dazu werden mit Hilfe der Eigenschaften der DFT die Algorithmen von Berlekamp bzw. Berlekamp-Massey in den Signalbereich übertragen. Dabei werden die Spektralvektoren der Länge t auf Signalvektoren der Länge n abgebildet, wodurch sich die Anzahl der im Zeitbereich zur Decodierung benötigten Operationen erhöht.

(3.18) Bemerkungen:

(a) Die Decodierung im Zeitbereich erfordert wie die Decodierung im Frequenzbereich $GF(q^m)$-Arithmetik.

(b) Zur Hardware-Implementierung der Decodierung im Zeitbereich mittels Berlekamp-Massey-Algorithmus benötigt man 7 Multiplizierer und einen Invertierer. Die Decodierung erfordert dann n^2 Takte (cf. /Fu84/).

(c) Zur Hardware-Implementierung der Decodierung im Zeitbereich mittels Berlekamp-Algorithmus benötigt man 13 Multiplizierer und einen Invertierer. Die Decodierung erfordert $2t \cdot n$ Takte (cf. /Fu84/).

IV. Algebraische DFT und ADFT-Codes

Wie im vorstehenden Kapitel deutlich wurde, nutzen viele Anwendungen (insbesondere auch aus dem Gebiet der fehlerkorrigierenden Codes) die volle Mächtigkeit der Diskreten Fourier-Transformation nicht aus. In diesen Fällen stellt die ***Algebraische Diskrete Fourier-Transformation*** (***ADFT***, cf. /Be83a/, /Be83b/, /Be84b/) das adäquate Verfahren zur Berechnung der DFT dar. Die Tatsache, daß die ADFT effizient implementiert werden kann (siehe §2), erhöht die Attraktivität der auf der DFT basierenden Codier- und Decodierverfahren beträchtlich.

Daneben kann die ADFT zur Definition einer interessanten neuen Klasse linearer Codes genutzt werden (§3). Die ADFT-Codes umfassen die Klasse der zyklischen Codes im engeren Sinn und besitzen wie diese die für die Implementierung wichtige Eigenschaft, daß die Syndromberechnung im wesentlichen mit derselben Hardware wie die Codierung durchgeführt werden kann (§4).

§1 Die Algebraische Diskrete Fourier-Transformation

(1.1) Definition:

Seien n und q natürliche Zahlen mit (n,q) = 1. Sei $GF(q^m)$ der Zerfällungskörper des Polynoms $x^n-1 \in GF(q)[x]$, α eine primitive n-te Einheitswurzel über GF(q) und $B = \langle b_0, b_1, \ldots, b_{m-1} \rangle$ eine GF(q)-Normalbasis von $GF(q^m)$. Ferner bezeichne ${}^BD_n = ((\alpha^{ij}))$ die n-te Fouriermatrix über GF(q), deren Einträge als GF(q)-Vektoren in der Basis B dargestellt sind, d.h. ${}^BD_n$ ist eine $(n \times n \cdot m)$-Matrix $((d_{i,j}))$ über GF(q). Dann heißt die $(n \times n)$-Matrix

$$ {}^BA_n = ((d_{i,m \cdot j}))_{0 \le i,j < n} $$

eine **n-te ADFT-Matrix** über GF(q). Für einen Vektor $\underline{u} \in GF(q)^n$ heißt der Vektor $\underline{U} = \underline{u}\,{}^BA_n$ **ADFT-Spektrum** von $\underline{u}$. Eine ADFT der Länge $n = q^m-1$ heißt **primitiv**.

(1.2) Bemerkungen:

(a) Für m = 1 stimmt die ADFT mit der DFT überein.

(b) ADFT-Matrizen sind nicht-singulär (cf. /Mü82/).

(c) Das DFT-Spektrum $\underline{u}\,{}^BD_n$ des Vektors $\underline{u}$ kann ohne Berechnungsaufwand aus dem ADFT-Spektrum von $\underline{u}$ erhalten werden (cf. /Mü82/).

(1.3) Beobachtung:

Eine ADFT-Matrix ist durch ihre erste Spalte $(d_{i,m})_{0\leq i<n}$ vollständig bestimmt. Für $0 \leq s < n$ erhält man die Spalte s als den Vektor $(d_{i\cdot s,m})_{0\leq i<n}$.

(1.4) Beispiel:

Sei $n = 7$ und $q = 2$. $GF(2^3)$ ist der Zerfällungskörper des Polynoms $x^7-1 \in GF(2)[x]$. Sei α Nullstelle von $x^3+x^2+1 \in GF(2)[x]$ und somit primitive 7-te Einheitswurzel über GF(2). Dann ist $B = \langle\alpha^4,\alpha^2,\alpha\rangle$ eine Normalbasis von GF(8) über GF(2) und

$$ {}^{B}A_7 = \begin{bmatrix} 1&1&1&1&1&1&1 \\ 1&0&0&1&1&1&0 \\ 1&0&1&0&0&1&1 \\ 1&1&0&0&1&0&1 \\ 1&1&0&1&0&0&1 \\ 1&1&1&0&0&1&0 \\ 1&0&1&1&1&0&0 \end{bmatrix} $$

eine 7-te ADFT-Matrix über GF(2).

Die *polynomiale Interpretation* der ADFT erlaubt die serielle Berechnung des ADFT-Spektrums mit linearen Rückkopplungs-Schieberegistern.

(1.5) Beobachtung (vgl. Beobachtung (III.1.2)):

Sei ${}^{B}A_n$ eine ADFT-Matrix über GF(q) und $\underline{u} = (u_0,u_1,\ldots,u_{n-1}) \in GF(q)^n$. Dann gilt für den i-ten ADFT-Spektralkoeffizienten

$$U_i = \Pi_0({}^{B}(u(x) \bmod (x-\alpha^i))) = \Pi_0({}^{B}u(\alpha^i))$$

für eine geeignete primitive n-te Einheitswurzel α über GF(q) (hierbei bezeichnet Π_i die i-te Komponentenfunktion, d.h. $\Pi_i(v_0,v_1,\ldots,v_{m-1}) = v_i$). Sei ferner $M_i(x)$ das Minimalpolynom von α^i. Die Reduktion des Signalpolynoms u(x) modulo $M_i(x)$ liefert

$$u(x) = q(x)\, M_i(x) + r(x)$$

und man erhält sofort

$$U_i = \Pi_0({}^{B}(r(\alpha^i))).$$

Für $B = <\beta_0,\beta_1,\ldots,\beta_{m-1}>$ mit $\beta_{i+1} = \beta_i^{\,q}$ gilt außerdem

$$U_{iq} = \Pi_0({}^{B}(r(\alpha^{iq}))) = \Pi_1({}^{B}(r(\alpha^i))).$$

Beobachtung (1.5) lehrt, daß die Berechnung der ADFT-Spektralkoeffizienten durch Reduktion des Signalpolynoms modulo der Minimalpolynome der n-ten Einheitswurzeln und Transformation der resultierenden Reste in die Normalbasis B erfolgen kann.

(1.6) Beobachtungen (vgl. Folgerung (III.1.5)):

(a) Ein Polynom p(x) aus dem Signalbereich besitzt α^i als Nullstelle genau dann, wenn für alle $j \in K(i,q)$ die ADFT-Spektralkoeffizienten P_j Null sind.

(b) Ein ADFT-Spektralpolynom $P(x)$ besitzt α^{-i} als Nullstelle genau dann, wenn für alle $j \in K(i,q)$ die Signalkoeffizienten p_j Null sind.

Im folgenden werden die in diesem Rahmen benötigten weiteren Eigenschaften der ADFT ohne Beweis angegeben. Für eine ausführlichere Darstellung sei auf /Mü82/, /Be83a/, /Be83b/, /Be84a/ und /Be84b/ verwiesen.

(1.7) Inversionseigenschaft:

Seien $B = \langle b_0, b_1, \ldots, b_{m-1} \rangle$ und $C = \langle c_0, c_1, \ldots, c_{m-1} \rangle$ spurorthogonale Normalbasen von $GF(q^m)$ über $GF(q)$ und sei

$$({}^{C}D_n)^{-1} = ((d_{i,j}))_{0 \le i < n, 0 \le j < n \cdot m}$$

die Inverse der n-ten Fouriermatrix über $GF(q)$, deren Einträge als $GF(q)$-Vektoren in der Basis C dargestellt sind. Dann gilt

$$({}^{B}A_n)^{-1} = ((d_{i,m \cdot j}))_{0 \le i,j < n} .$$

Ist B eine selbst-duale Normalbasis, so heißt die Matrix ${}^{B}A_n$ *ADFT-Matrix mit Inversionseigenschaft*.

(1.8) **Bemerkung**:

Mit (1.7) und der Inversionseigenschaft der DFT erhält man die polynomiale Interpretation der inversen ADFT. Für den i-ten ADFT-Signalkoeffizienten eines ADFT-Spektrums $\underline{U} = (U_0, U_1, \ldots, U_{n-1}) \in GF(q)^n$ gilt

$$u_i = 1/n \; \Pi_0(^C U(\alpha^{-i}))$$

für eine geeignete primitive n-te Einheitswurzel α über GF(q).

(1.9) **Beispiel**:

Sei q = 2 und n = 9. $GF(2^6)$ ist der Zerfällungskörper des Polynoms $x^9-1 \in GF(2)[x]$. Sei α Nullstelle von $x^6+x+1 \in GF(2)[x]$ und somit primitive 63-te Einheitswurzel über GF(2). Dann bildet die Konjugiertenklasse von α^{23} eine selbstduale Normalbasis B von GF(64) über GF(2) (siehe Anhang (A1)) und

$$B_{A_9} = \begin{bmatrix} 1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&0&0&0 \\ 1&1&1&0&0&1&1&1&0 \\ 1&1&0&1&1&0&1&1&0 \\ 1&1&0&1&0&1&0&1&1 \\ 1&1&1&0&1&0&1&0&1 \\ 1&0&1&1&0&1&1&0&1 \\ 1&0&1&1&1&0&0&1&1 \\ 1&0&0&0&1&1&1&1&1 \end{bmatrix}$$

ist eine binäre 9-te ADFT-Matrix mit Inversionseigenschaft. Weiter sind $\beta = \alpha^7$ und $\gamma = \alpha^{21}$ primitive 9-te bzw. 3-te Einheitswurzeln mit den Minimalpolynomen x^6+x^3+1 und x^2+x+1.

(1.10) Phaseneigenschaft:

Für die ADFT-Spektralkoeffizienten des zu einem Signalvektor $\underline{v} = (v_0, v_1, \ldots, v_{n-1})$ zyklisch verschobenen Signals $\underline{v}' = (v_d, v_{d+1}, \ldots, v_{d+n-1})$ gilt:

$$V_j' = \Pi_0({}^{B}(\alpha^{-jd} \cdot v(\alpha^j))).$$

(1.11) Skalierungseigenschaft:

Für die ADFT-Spektralkoeffizienten des gegenüber einem Signalvektor $\underline{v} = (v_0, v_1, \ldots, v_{n-1})$ durch einen Skalierungsfaktor $s > 0$ mit $(n,s) = 1$ permutierten Signals $\underline{v}' = (v_0, v_s, \ldots, v_{(n-1)s})$ gilt:

$$V_j' = V_{j/s}.$$

(1.12) Faltungseigenschaft:

Das ADFT-Spektrum $\underline{V}$ der zyklischen Faltung zweier Signalvektoren $\underline{u}$ und $\underline{w}$ kann wie folgt berechnet werden:

$$V_j' = \Pi_0({}^{B}(u(\alpha^j) \cdot w(\alpha^j))).$$

Aus naheliegenden Gründen sind für die Praxis ADFT-Matrizen mit Inversionseigenschaft besonders interessant.

(1.13) Satz:

Besitzt eine ADFT-Matrix A_n über GF(q) die Inversionseigenschaft und gilt $n = s \cdot d$, so gibt es eine ADFT-Matrix A_s über GF(q) mit Inversionseigenschaft.

Beweis: Bezeichne $\underline{a} = (a_0, a_1, \ldots, a_{n-1})$ Spalte 1 der Matrix A_n und $\underline{b} = (b_0, b_1, \ldots, b_{s-1}) := (a_0, a_d, \ldots, a_{(s-1)d})$ den durch Abtasten von $\underline{a}$ mit Schrittweite d erhaltenen Vektor. Ist p die Charakteristik von GF(q), so kann die Inversionseigenschaft von A_n für $0 \leq i,j \leq n-1$ ausgedrückt werden durch

$$\sum_{k=0}^{n-1} a_{ik} \cdot a_{jk} \equiv \begin{cases} n \bmod p, & \text{falls } i+j \equiv 0 \bmod n \\ 0 \bmod p & \text{sonst.} \end{cases}$$

Betrachte nun für $0 \leq i,j \leq s-1$

$$\sum_{k=0}^{s-1} b_{ik} \cdot b_{jk} = \sum_{k=0}^{s-1} a_{idk} \cdot a_{jdk} = 1/d \sum_{k=0}^{n-1} a_{idk} \cdot a_{jdk} \equiv$$

$$\equiv \begin{cases} s \bmod p, & \text{falls } id+jd \equiv 0 \bmod n \\ 0 \bmod p & \text{sonst} \end{cases}$$

Offenbar gilt $id+jd \equiv 0 \bmod n$ genau dann, wenn $i+j \equiv 0 \bmod s$ und somit ist der Vektor $\underline{b}$ Spalte 1 einer ADFT-Matrix A_s mit Inversionseigenschaft.

□

Abschließend soll der für viele Anwendungen wichtigste Fall $q = 2$ näher betrachtet werden. Sei $GF(2^m)$ der Zerfällungskörper des Polynoms $x^n-1 \in GF(2)[x]$. Für $m \equiv 1 \bmod 2$ ist die Anzahl der GF(2)-Normalbasen von $GF(2^m)$ ungerade (cf. /Ma78/) und mit Bemerkung (I.1.14b) folgt, daß in diesem Fall eine binäre n-te ADFT-Matrix mit Inversionseigenschaft existiert (siehe auch /Be83a/ und /Zi83/). Für $m \equiv 2 \bmod 4$ wurde von K.H.Zimmermann /Zi83/ ein konstruktives Verfahren zur Bestimmung einer selbst-dualen Normalbasis von $GF(2^m)$ über GF(2) angegeben. Den Fall $m \equiv 0 \bmod 4$ schließlich klärt der folgende Satz.

(1.14) Satz:

Es existiert keine binäre ADFT-Matrix A_n mit Inversionseigenschaft, falls der Erweiterungsgrad m des Zerfällungskörpers $GF(2^m)$ von $x^n-1 \in GF(2)[x]$ kongruent 0 modulo 4 ist.

Beweis: Durch Induktion zeigt man leicht, daß gilt

$$m \equiv 0 \bmod 4 \iff 2^m-1 \equiv 0 \bmod 5.$$

Angenommen, für $m \equiv 0 \bmod 4$ und $n = 2^m-1$ existiert eine ADFT-Matrix A_n über GF(2) mit Inversionseigenschaft. Dann existiert nach Satz (1.11) eine binäre ADFT-Matrix A_5 mit Inversionseigenschaft. Sei $\underline{a} = (a_0, a_1, a_2, a_3, a_4)$ Spalte 1 der Matrix A_5. Dann ist $\underline{a}' = (a_0, a_2, a_4, a_1, a_3)$ Spalte 2 von

A_5 und es gilt $a_0 = 1$ und $wgt(\underline{a}) \equiv 0 \bmod 2$. Damit folgt $\underline{a} \cdot \underline{a}' \equiv 1 \bmod 2$ im Widerspruch zur Annahme.

(1.15) **Bemerkung** (cf. /Be84b/):

Im allgemeinen Fall existiert über einem Körper GF(q) genau dann eine n-te ADFT-Matrix mit Inversionseigenschaft, wenn für den Erweiterungsgrad m des Zerfällungskörpers $GF(q^m)$ von $x^n-1 \in GF(q)[x]$ gilt:

$$m \neq \begin{cases} 0 \bmod 4, \text{ falls } q \equiv 0 \bmod 2 \\ 0 \bmod 2 \quad \text{sonst} . \end{cases}$$

In untenstehender Tabelle sind diejenigen Zahlen n angegeben, für die der Erweiterungsgrad m des Zerfällungskörpers $GF(2^m)$ von $x^n-1 \in GF(2)[x]$ höchstens 12 ist. In der Spalte "IE" ist die Existenz einer binären n-ten ADFT-Matrix mit Inversionseigenschaft gekennzeichnet.

n	m	IE	n	m	IE	n	m	IE
3	2	ja	39	12	-	195	12	-
5	4	.	45	12	-	255	8	-
7	3	ja	51	8	-	273	12	-
9	6	ja	63	6	ja	315	12	-
11	10	ja	65	12	-	341	10	ja
13	12	-	73	9	ja	455	12	-
15	4	-	85	8	-	511	9	ja
17	8	-	89	11	ja	585	12	-
21	6	ja	91	12	-	819	12	-
23	11	ja	93	10	ja	1023	10	ja
31	5	ja	105	12	-	1365	12	-
33	10	ja	117	12	-	2047	11	ja
35	12	-	127	7	ja	4095	12	-

(1.16) Tabelle

§2 Zur Implementierung der ADFT

Häufig ist für die praktische Bedeutung einer Transformation die Verfügbarkeit effizienter Implementierungen entscheidend. In diesem Abschnitt werden daher die wesentlichen Methoden zur (Hardware-) Realisierung einer ADFT der Länge n über GF(q) betrachtet und im Hinblick auf ihren Implementierungsaufwand untersucht:

(M1) die direkte (parallele) Durchführung der linearen Transformation $\underline{U} = \underline{u}^{B}A_{n}$;

(M2) die auf der polynomialen Interpretation der ADFT basierende serielle Implementierung der Transformation mittels linearer Rückkopplungs-Schieberegister;

(M3) die Zerlegung einer ADFT der Länge $n = r \cdot s$ in s Transformationen der Länge r und r Transformationen der Länge s, die ihrerseits mit der Methode (M1), (M2) oder (M3) implementiert werden können.

(2.1) Beobachtung:

Zur parallelen Implementierung der Transformation $\underline{U} = \underline{u}\,{}^{B}A_{n}$ werden n^2 Multiplizierer mit Konstanten und $n(n-1)$ Addierer benötigt. Im Fall $q = 2$ entfällt der Aufwand für die Multiplikationen, so daß eine binäre ADFT der Länge n mit weniger als n^2 XOR-Gattern in $O(\log(n))$ Gatterlaufzeiten berechnet werden kann.

Für eine binäre, primitive ADFT mit Primzahl-Länge kann die obige Abschätzung verbessert werden:

(2.2) Satz:

Sei ${}^{B}A_{n}$ eine n-te ADFT-Matrix über GF(2) und $n = 2^m-1$ eine Primzahl. Dann werden zur parallelen Implementierung der Transformation $\underline{U} = \underline{u}\,{}^{B}A_{n}$ höchstens $(2n-m)(n-1)/m$ XOR-Gatter benötigt.

Beweis: Da $n = 2^m-1$ Primzahl ist, ist jedes $\alpha \in GF(2^m)^*$, $\alpha \neq 1$ primitive n-te Einheitswurzel über GF(2), d.h. für $i \neq 0$ bilden die Spalten $i, 2\cdot i, 4\cdot i, \ldots, 2^{m-1}\cdot i$ der ADFT-Matrix ${}^{B}A_{n}$ die Kontrollmatrix H eines binären (n,n-m)-Hamming-Codes. Die durch H beschriebene Abbildung kann mit $2(n-m)$ XOR-Gattern implementiert werden (cf. /Sa78/). Zur parallelen Berechnung der gesamten ADFT sind demnach

$$((n-1)/m)(2(n-m))+(n-1) = ((n-1)/m)(2n-m)$$

XOR-Gatter ausreichend.

□

In Tabelle (2.3) sind für einige Transformationslängen die Zahlen der für

- die direkte (parallele) Berechnung der DFT,
- die Berechnung der DFT mit Hilfe des effizienten Verfahrens von D.V.Sarwate /Sa78/ und
- die Berechnung der ADFT unter Zugrundelegung der in Satz (2.2) gegebenen Schranke

jeweils notwendigen Bit-Operationen gegenübergestellt (cf. Tabelle 1 in /Sa78/).

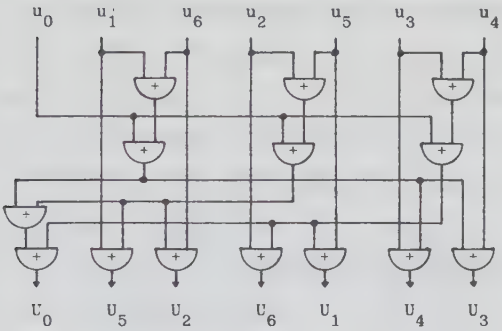
Vorgehensweise	n = 7	n = 31	n = 127
direkte Berechnung der DFT	216	12000	395136
schnelle Berechnung der DFT	126	2790	39438
Berechnung der ADFT nach (2.2)	22	342	4410

(2.3) Tabelle

Der tatsächlich nötige Aufwand für die parallele Implementierung der ADFT unterschreitet die Schranke aus Satz (2.2) in vielen Fällen deutlich (siehe auch Anhang (A2)).

(2.4) **Beispiel**:

Die durch die 7-te ADFT-Matrix über GF(2) aus Beispiel (1.6) beschriebene Transformation kann durch die in Abbildung (2.5) dargestellte Schaltung mit 14 XOR-Gattern in 4 Gatterlaufzeiten parallel berechnet werden.



(2.5) **Abbildung**: Parallele Implementierung einer binären ADFT der Länge 7

(2.6) Beobachtung:

Zu der auf der polynomialen Interpretation basierenden seriellen Implementierung der ADFT werden für die Rückkopplungs-Schieberegister n GF(q)-Speicherzellen, sowie höchstens n Addierer und Multiplizierer mit Konstanten benötigt. Der Aufwand für die Basistransformationen kann für eine primitive ADFT (d.h. $n = q^m-1$) mit $O(n \cdot m)$ GF(q)-Operationen abgeschätzt werden. Für eine nicht-primitive ADFT können im ungünstigsten Fall ($n = m+1$) $O(n^2)$ Operationen nötig sein. Eine primitive ADFT über GF(2) kann somit in n Takten mit einer Schaltung aus n Flip-Flops und $O(n \cdot \log(n))$ XOR-Gattern berechnet werden.

(2.7) Beispiel (vgl. Beispiel (2.4)):

Sei $q = 2$ und $n = 7$. Das Polynom $x^7-1 \in GF(2)[x]$ zerfällt wie folgt in Minimalpolynome:

$$x^7-1 = (x-1)\cdot(x^3+x^2+1)\cdot(x^3+x+1) = M_0(x)\cdot M_1(x)\cdot M_3(x).$$

Sei α primitive 7-te Einheitswurzel über GF(2) und $M_1(x)$ das Minimalpolynom von α. Dann ist $B = \langle \alpha^4, \alpha^2, \alpha \rangle$ eine (selbstduale) Normalbasis von $GF(2^3)$ über GF(2). Zu einem Signalvektor $\underline{u} \in GF(2)^7$ wird durch die in Abbildung (2.8) dargestellten Divisions-Schieberegister in 7 Takten berechnet:

$u(x) \bmod M_0(x)$ in der Basis $\langle 1 \rangle$,

$u(x) \bmod M_1(x)$ in der Basis $\langle 1, \alpha, \alpha^2 \rangle$ und

$u(x) \bmod M_3(x)$ in der Basis $\langle 1, \alpha^3, \alpha^6 \rangle$.

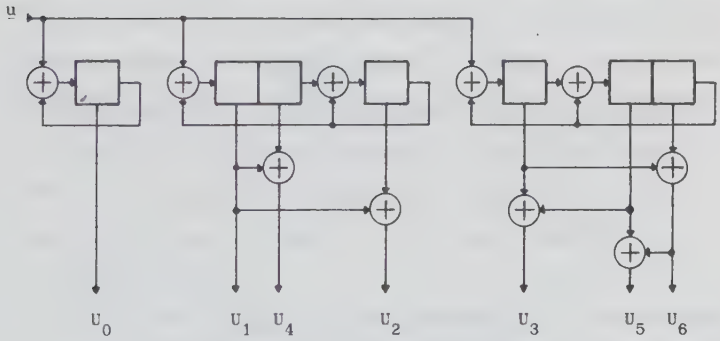
Die in den verschiedenen polynomialen Basen dargestellten Reste werden mit den im unteren Teil der Schaltung realisierten Basis-Transformationen durch Ausnutzen der Beziehungen

$$1 = \alpha + \alpha^2 + \alpha^4,$$

$$\alpha^3 = \alpha + \alpha^4 \quad \text{und}$$

$$\alpha^6 = \alpha + \alpha^2$$

in die (gemeinsame) Normalbasis B transformiert (vgl. Beobachtung (1.5)).



(2.8) Abbildung: Serielle Implementierung einer binären ADFT der Länge 7

Die Faktorisierungseigenschaften der Fourier-Matrix bei nicht-primer Transformationslänge n führen zu den bekannten "schnellen" Berechnungsverfahren für die DFT (siehe z.B. /Nu81/, /Be84/). Analoge Algorithmen können auch für die ADFT formuliert werden. An dieser Stelle soll nur das dem *Good-Thomas-Algorithmus* für die DFT (der mißverständlich auch als Primfaktor-Algorithmus bezeichnet wird, cf. /Go58/, /Th63/) entsprechende Faktorisierungsverfahren für die ADFT näher betrachtet werden.

(2.9) **Beobachtung:**

Seien r und s natürliche Zahlen mit (r,s) = 1 und sei n = r·s. Dann gilt

$$\mathbb{Z}_r \times \mathbb{Z}_s = \mathbb{Z}_n .$$

Dieser (als *chinesischer Restesatz* bekannte) Ringisomorphismus wird durch die Abbildung

$$\rho : (x,y) \dashrightarrow (s\cdot s'\cdot x + r\cdot r'\cdot y) \bmod n$$

vermittelt, wobei $s\cdot s' \equiv 1 \bmod r$ und $r\cdot r' \equiv 1 \bmod s$ gelte. Dann ist

$$\rho^{-1} : z \dashrightarrow (z \bmod r,\ z \bmod s)$$

und die eindimensionale DFT eines Signalvektors $\underline{u} \in GF(q)^n$ kann als zweidimensionale Fourier-Transformation der Signalmatrix $v = ((u_{\rho(i,j)}))_{0\le i<r,\,0\le j<s}$ berechnet werden (cf. /Nu81/, /Mü82/, /Be84/). Die Beziehung

$$U_k = V_{k\cdot s' \bmod r,\ k\cdot r' \bmod s}$$

beschreibt die resultierende Anordnung des Spektrums.

Die Frage nach der Übertragbarkeit dieser Zerlegung auf die ADFT wird durch die folgende Beobachtung beantwortet:

(2.10) Beobachtung (cf. /Zi83/):

Seien a und b natürliche Zahlen mit (a,b) = 1, sowie $A = \langle\alpha_0,\alpha_1,\ldots,\alpha_{a-1}\rangle$ und $B = \langle\beta_0,\beta_1,\ldots,\beta_{b-1}\rangle$ GF(q)-Basen von $GF(q^a)$ bzw. $GF(q^b)$. Dann ist $C = \langle\alpha_i\cdot\beta_j\rangle_{0\leq i<a,0\leq j<b}$ eine Basis von $GF(q^{a\cdot b})$ über GF(q). Sind A und B Normalbasen, so ist auch C eine Normalbasis. Sind A und B darüber hinaus selbst-dual, so ist auch C selbst-dual.

Eine ADFT der Länge $n = r\cdot s$ (mit (r,s) = 1) kann somit ohne weiteren Aufwand in s Transformationen der Länge r und r Transformationen der Länge s zerlegt werden, falls die Erweiterungsgrade der Zerfällungskörper von x^r-1 und x^s-1 über GF(q) relativ prim sind.

(2.11) Bemerkungen:

(a) Bei parallerer Implementierung der beschriebenen Faktorisierung einer ADFT der Länge $n = r\cdot s$ ergibt sich der nötige Aufwand als

$$s\cdot(\text{ Aufwand für ADFT der Länge } r\ ) + r\cdot(\text{ Aufwand für ADFT der Länge } s\ )$$

Das ADFT-Spektrum wird in der Summe der Zeiten für die Transformation der Länge r und die der Länge s berechnet.

(b) Bei serieller Implementierung der beschriebenen Faktorisierung einer ADFT der Länge $n = r \cdot s$ ergibt sich der nötige Aufwand als Summe des Aufwands für die Transformation der Länge r und die der Länge s. Zusätzlich wird ein Zwischenspeicher der Größe n benötigt. Das ADFT-Spektrum wird in

$$s \cdot (\text{ Zeitaufwand für ADFT der Länge } r) + r \cdot (\text{ Zeitaufwand für ADFT der Länge } s)$$

berechnet.

(2.12) Beispiel:

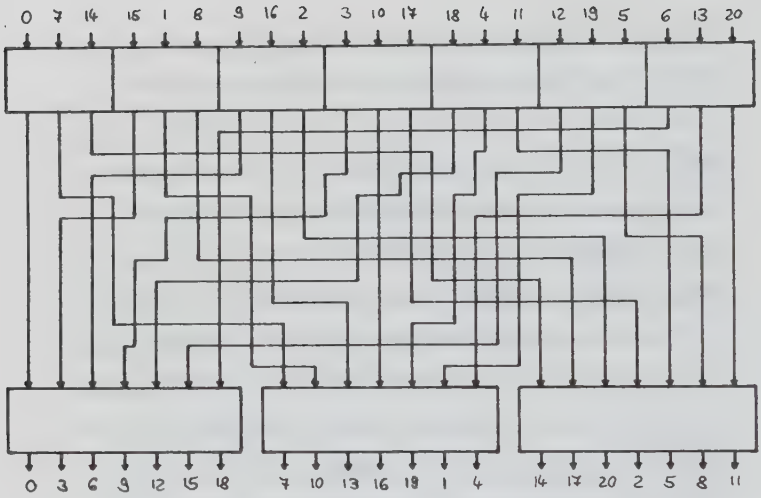
Seien q = 2, r = 3 und s = 7. Dann ist r' = 5 und s' = 1 und die folgenden Tabellen geben den Isomorphismus ρ sowie die resultierende Anordnung der Spektralkoeffizienten an.

ρ	0	1	2	3	4	5	6
0	0	15	9	3	18	12	6
1	7	1	16	10	4	19	13
2	14	8	2	17	11	5	20

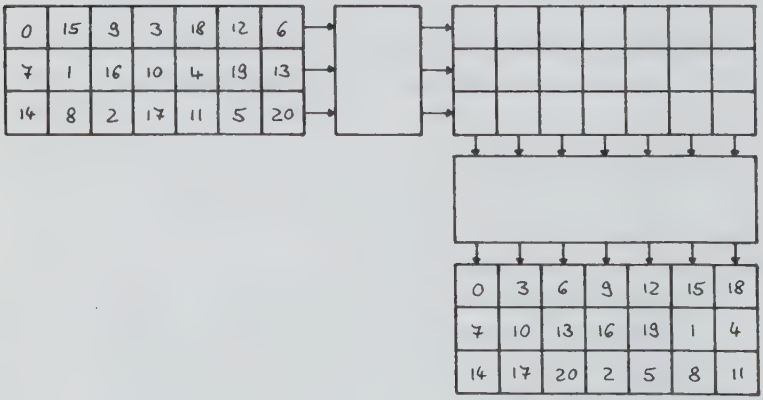
σ	0	1	2	3	4	5	6
0	0	3	6	9	12	15	18
1	7	10	13	16	19	1	4
2	14	17	20	2	5	8	11

Sei nun γ eine primitive 21-ste Einheitswurzel über GF(2). Dann sind $\alpha = \gamma^3$ und $\beta = \gamma^7$ primitive 7-te bzw. 3-te Einheitswurzeln. Seien weiter x^3+x^2+1 und x^2+x+1 die Minimalpolynome von α bzw. β. Dann sind $A = \langle \alpha^4, \alpha^2, \alpha \rangle$ und

$B = <\beta, \beta^2>$ selbst-duale GF(2)-Normalbasen von $GF(2^3)$ bzw. $GF(2^2)$. Mit Beobachtung (2.10) erhält man aus A und B die selbst-duale Normalbasis $C = <\gamma^{19}, \gamma^{20}, \gamma^{10}, \gamma^5, \gamma^{13}, \gamma^{17}>$ von $GF(2^6)$ über GF(2). In den Abbildungen (2.13) und (2.14) sind die zugehörigen Implementierungen einer binären ADFT der Länge 21 dargestellt. Bei paralleler Realisierung der Transformationen der Längen 3 und 7 kann die ADFT der Länge 21 im Fall der parallelen Implementierung der Faktorisierung mit $63 = 7 \cdot 3 + 3 \cdot 14$ XOR-Gattern in $6 = 2 + 4$ Gatterlaufzeiten berechnet werden. Bei der seriellen Implementierung der Faktorisierung sind $17 = 3 + 14$ XOR-Gatter, 21 Flip-Flops und $10 = 3 + 7$ Takte erforderlich. Zum Vergleich: Die parallele Implementierung einer ADFT der Länge 21 ist mit 114 XOR-Gattern möglich (siehe Anhang (A2)).



(2.13) **Abbildung**: Faktorisierung einer ADFT der Länge 21 (parallel)



(2.14) **Abbildung**: Faktorisierung einer ADFT der Länge 21 (seriell)

(2.15) **Bemerkung**:

Eine ADFT der Länge $n = r \cdot s$ (mit $(r,s) = 1$) über GF(q) kann auch dann in s Transformationen der Länge r und r Transformationen der Länge s zerlegt werden, wenn die Erweiterungsgrade der Zerfällungskörper von x^r-1 und x^s-1 über GF(q) nicht relativ prim sind. Allerdings kann in dieser Situation das gewünschte eindimensionale ADFT-Spektrum nicht ohne zusätzlichen Aufwand aus dem zweidimensionalen ADFT-Spektrum gewonnen werden (cf. /Be84a/).

§3 ADFT-Codes

Ein linearer Code der Länge n über einem Alphabet GF(q) kann durch die ADFT-Spektren seiner Codewörter beschrieben werden, falls n und q relativ prim sind und somit eine ADFT zu den Parametern n und q existiert. Analog zur Beschreibung zyklischer Codes durch die DFT kann mit Hilfe der ADFT eine interessante neue Klasse linearer Codes definiert werden.

(3.1) Definition:

Sei $A_n = (\underline{a}_0, \underline{a}_1, \ldots, \underline{a}_{n-1})$ eine n-te ADFT-Matrix über dem Körper GF(q) und sei $Z = \{z[0], z[1], \ldots, z[s-1]\}$ eine Teilmenge von [0:n-1] der Mächtigkeit s. Dann ist der durch die Kontrollmatrix

$$H = (\underline{a}_{z[0]}, \underline{a}_{z[1]}, \ldots, \underline{a}_{z[s-1]})^t$$

definierte lineare Code *C* ein *ADFT-Code*. Ein ADFT-Code der Länge $n = q^m - 1$ heißt *primitiv*.

(3.2) Bemerkungen:

(a) Für das ADFT-Spektrum $\underline{C}$ jedes Codewortes $\underline{c}$ eines ADFT-Codes *C* gilt:

$$i \in Z \implies C_i = 0$$

Die Elemente von Z werden daher als die *Kontrollfrequenzen* des Codes *C* bezeichnet.

(b) Jedes ADFT-Spektrum $\underline{C}$ mit

$$C_j = 0 \qquad \text{für } j \in Z$$

ist ein *Codewort im Frequenzbereich*.

Die folgenden Beobachtungen ergeben sich unmittelbar aus den Eigenschaften der ADFT.

(3.3) Beobachtungen:

(a) C ist ein (n,n-s)-Code.

(b) Sei $U = \{u_0, u_1, \ldots, u_{n-s-1}\} := [0:n-1]\backslash Z$ und sei $B_n = (\underline{b}_0, \underline{b}_1, \ldots, \underline{b}_{n-1}) = A_n^{-1}$. Dann ist

$$G = (\underline{b}_{u[0]}, \underline{b}_{u[1]}, \ldots, \underline{b}_{u[n-s-1]})^t$$

eine Generatormatrix von C. Besitzt A_n die Inversionseigenschaft, so gilt

$$G = (\underline{a}_{-u[0]}, \underline{a}_{-u[1]}, \ldots, \underline{a}_{-u[n-s-1]})^t.$$

(c) Das orthogonale Komplement $C^\perp$ eines (n,k)-ADFT-Codes C ist ein (n,n-k)-ADFT-Code. Die Kontrollfrequenzen von $C^\perp$ sind die Inversen (bzgl. der Addition mod n) der Nicht-Kontrollfrequenzen von C.

(3.4) Beispiel:

Sei $q = 2$, $n = 7$, A_7 die 7-te ADFT-Matrix mit Inversionseigenschaft aus Beispiel (1.4) und sei $Z = \{1,2,4\}$. Dann ist

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Kontrollmatrix und

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Generatormatrix eines binären (7,4)-ADFT-Codes, der die Minimaldistanz d = 3 besitzt (Satz I.1.8) und äquivalent zum (7,4)-Hamming-Code ist.

(3.5) Beobachtungen:

(a) Sind Z_1 und Z_2 die Mengen der Kontrollfrequenzen zweier mit Hilfe derselben ADFT-Matrix definierter ADFT-Codes C_1 und C_2 der Länge n über GF(q) und gilt $Z_2 \subset Z_1$, so ist C_1 Teilmenge von C_2.

(b) Ist A_n eine ADFT-Matrix mit Inversionseigenschaft und Z die Menge der Kontrollfrequenzen des (n,k)-ADFT-Codes C über GF(q) derart, daß

$$\{ x \in [0:n-1] \mid -x \notin Z \} \subset Z ,$$

so ist C schwach selbst-dual.

(3.6) Beispiel:

Sei $q = 2$, $n = 7$, A_7 die 7-te ADFT-Matrix mit Inversionseigenschaft aus Beispiel (1.4) und sei $Z = \{0,1,2,4\}$. Dann ist

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Generatormatrix eines schwach selbst-dualen binären (7,3)-ADFT-Codes mit Minimaldistanz 4, der zu dem (7,4)-ADFT-Code aus Beispiel (3.4) dual ist.

(3.7) Satz:

Ein (n,k)-ADFT-Code über GF(q) ist genau dann zyklisch, wenn die ihn definierende Menge der Kontrollfrequenzen Vereinigung von Kreisteilungsklassen modulo n bezüglich q ist.

Beweis:

Die Behauptung folgt unmittelbar aus Beobachtung (1.6).

□

(3.8) Folgerung:

Jeder zyklische Code im engeren Sinn (d.h. jeder zyklische (n,k)-Code über GF(q) mit (n,q) = 1) ist ein ADFT-Code.

Aus der Beschreibung einer Klasse von Codes mit Hilfe der DFT (siehe Tabelle (III.2.7)) kann unmittelbar eine Charakterisierung dieser Codeklasse durch die ADFT gewonnen werden. Jedoch sind nicht alle derart beschreibbaren Codes ADFT-Codes. Die bedeutendsten Teilklassen der Klasse der ADFT-Codes sind in Tabelle (3.9) zusammengestellt (vgl. Tabelle (III.2.7)).

Code	Länge	Kontroll-frequenzen	davon konsekutiv	Neben-bedingungen
ADFT	(n,q)=1	n-k	-	nein
zyklisch	(n,q)=1	n-k	-	ja
BCH	(n,q)=1	n-k	d'-1	ja
RS	n|(q-1)	d-1	d-1	nein

(3.9) Tabelle: Codeklassen und ihre Beschreibung durch die ADFT

Mit Hilfe einer ADFT-Matrix ${}^{B}A_{n}$ können $\binom{n}{k}$ verschiedene ADFT-Codes der Dimension k ($0 \leq k \leq n$) definiert werden. Eine hinreichende Bedingung für die Äquivalenz von ADFT-Codes liefert:

(3.10) Satz:

Sei C ein (n,k)-ADFT-Code über GF(q), Z die Menge der Kontrollfrequenzen von C und

$$U = [0:n-1] \setminus Z = \{u_0, u_1, \ldots, u_{k-1}\}.$$

Dann ist für jedes d mit (n,d) = 1 der durch U' = d·U definierte ADFT-Code C' äquivalent zu C.

Beweis: Sei s keine Kontrollfrequenz von C und $\underline{a}_s$ der zugehörige Zeilenvektor der ADFT-Matrix. Der zu d·s (mod n) gehörende Zeilenvektor $\underline{a}_{ds}$ unterscheidet sich von $\underline{a}_s$ wegen (n,d) = 1 durch eine Permutation P_d der Komponenten.

□

(3.11) Bemerkung:

Äquivalente Codes sind nicht unbedingt auch für die Praxis als gleichwertig einzustufen. Zum einen können sie sich im Implementierungsaufwand (etwa der Anzahl der für die Syndromberechnung nötigen Rückkopplungen) unterscheiden (siehe auch §4), zum anderen in der Fähigkeit, bestimmte Fehlermuster korrigieren zu können. Letzteres kann unter der durchaus praxisnahen Annahme von Bedeutung sein, daß nicht alle Fehlermuster eines Gewichts mit gleicher Wahrscheinlichkeit auftreten. Beispielsweise wird das Auftreten von

Fehlerbündeln in vielen Anwendungen als wahrscheinlicher angesehen als das zufällig verteilter Fehlermuster gleichen Gewichts (siehe z.B. /Fu84/).

Die Kenntnis der Symmetrie-Eigenschaften eines Codes kann z.B. bei der Bestimmung seiner Minimaldistanz oder für seine Decodierung nützlich sein.

(3.12) Beobachtung:

Sei C ein (n,k)-ADFT-Code, Z die Menge der Kontrollfrequenzen von C und gelte für eine natürliche Zahl t mit $1 \leq t < n$ und $(n,t) = 1$

$$\{ t \cdot z \bmod n \mid z \in Z \} = Z,$$

so ist die Permutation der Indizes

$$\tau : \; i \dashrightarrow t \cdot i \bmod n$$

von [0:n-1] ein Automorphismus von C.

Die Minimaldistanz eines (n,k)-ADFT-Codes C über GF(q) ist von der Wahl der Kontrollfrequenzen und von der zugrundeliegenden ADFT-Matrix ${}^{B}A_{n}$ abhängig. Ist C ein zyklischer ADFT-Code so ist seine Minimaldistanz von der Basis B unabhängig.

(3.13) Beispiel:

Sei $q = 2$ und $n = 5$. $GF(2^4)$ ist der Zerfällungskörper des Polynoms x^5-1. Sei α eine primitive 15-te Einheitswurzel über GF(2) mit Minimalpolynom x^4+x+1. Dann sind $B = \langle\alpha^3,\alpha^6,\alpha^{12},\alpha^9\rangle$ und $C = \langle\alpha^7,\alpha^{14},\alpha^{13},\alpha^{11}\rangle$ GF(2)-Normalbasen von GF(16) (vgl. Anhang (A1)) und es gilt

$$ {}^{B}A_5 = \begin{bmatrix} 1&1&1&1&1\\ 1&1&0&0&0\\ 1&0&0&1&0\\ 1&0&1&0&0\\ 1&0&0&0&1 \end{bmatrix} \quad \text{und} \quad {}^{C}A_5 = \begin{bmatrix} 1&1&1&1&1\\ 1&1&1&0&1\\ 1&1&1&1&0\\ 1&0&1&1&1\\ 1&1&0&1&1 \end{bmatrix}. $$

Offenbar besitzt jeder mit Hilfe von ${}^{B}A_5$ definierte ADFT-Code der Dimension $k = 2$ die Minimaldistanz $d = 2$, wogegen ein mit Hilfe von ${}^{C}A_5$ definierter (5,2)-ADFT-Code eine Minimaldistanz $d \in \{1,2\}$ besitzt.

Im weiteren sei $\delta_{max}(n,k) = \max\{\, d \mid d \text{ Minimaldistanz eines binären } (n,k)\text{-Codes} \,\}$ und $d_{max}(n,k) = \max\{\, d \mid d \text{ Minimaldistanz eines binären } (n,k)\text{-ADFT-Codes} \,\}$.

(3.14) Satz:

Sei $n = 2^m-1$. Dann gilt für $2 \leq k \leq m$

$$ d_{max}(n,k) \begin{cases} = 2^{m-1}, & \text{falls } n \text{ Primzahl} \\ \geq 2^{m-1} & \text{sonst.} \end{cases} $$

Beweis: Sei $n = 2^m-1$. In einer primitiven binären ADFT-Matrix ${}^{B}A_n$ bilden die Spalten $1,2,4,\ldots,2^{m-1}$ unabhängig von B die Kontrollmatrix H eines $(n,n-m)$-Hamming-Codes. H ist andererseits Generatormatrix eines (n,m)-Simplex-Codes mit Minimaldistanz 2^{m-1}. Somit folgt für $2 \leq k \leq m$

$$d_{max}(n,k) \geq 2^{m-1}.$$

Ist $n = 2^m-1$ Primzahl, so besitzen in der ADFT-Matrix ${}^{B}A_n$ über GF(2) unabhängig von B alle Spalten s mit $0 < s < n$ das Gewicht 2^{m-1} (die Spalte 0 besitzt das Gewicht n). Daraus folgt $d_{max}(n,k) \leq 2^{m-1}$ für $k > 1$ und mit Teil 1 des Beweises die Behauptung.

□

Umfangreiche Tabellen der "besten" bekannten Blockcodes existieren (siehe z.B. Ma78). Bei festem Alphabet, gegebener Länge n und Minimaldistanz d wird derjenige Code als besser angesehen, der die kleinere Redundanz

$$R = n - \log(\text{Anzahl der Codeworte})$$

besitzt. In der Praxis dürfte daneben der Implementierungsaufwand für die Codierung und vor allem für die Decodierung meist eines der wichtigsten Kriterien darstellen.

Für den Bereich $3 \leq n \leq 23$ und $2 \leq k < n$ wurden die besten binären (n,k)-ADFT-Codes durch eine Computer-Suche ermittelt (siehe Anhang (A3)). Ein Vergleich der erhaltenen Resultate mit einer von H.J.Helgert und R.D.Stinaff angegebenen Tabelle (cf. /He73/) zeigt, daß für relativ viele Paare (n,k) die für binäre lineare Codes maximal mögliche Minimaldistanz $\delta_{max}(n,k)$ erreicht wird (Eintrag 'a' oder 'z' in Tabelle (3.15)). In dem systematisch untersuchten Bereich sind ca. 45% der ADFT-Codes mit Minimaldistanz $d = \delta_{max}(n,k)$ nicht zyklisch (Eintrag 'a' in Tabelle (3.15)). Für $n \geq 23$ sind in zunehmend vielen Fällen nur Schranken für $\delta_{max}(n,k)$ bekannt (cf. /He73/).

n\k	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
3	z	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
5		a	z	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7	a	z	z	a	z	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
9	z	a	a	a	a	z	z	.	.	.	.	.	.	.	.	.	.	.	.	.	.
11			-	a	-	-	a	a	z	.	.	.	.	.	.	.	.	.	.	.	.
13	-		a	-	a	a	-	-	a	a	z	.	.	.	.	.	.	.	.	.	.
15	z	a	z	z	z	z	z	z	z	z	a	z	z	.	.	.	.	.	.	.	.
17			a	-	-	a	z	z	a		-	a	a	a	z	.	.	.	.	.	.
21	z	z	a	z	z	z	a	z	-	z	z	z	z	z	z	a	z	z	z	.	.
23		a	-	-	-	?	a	a	a	z	z	-	-	-			-	a	a	a	z

(3.15) Tabelle: Häufigkeit "guter" ADFT-Codes für $n \leq 23$

Die Einträge 'a' in Tabelle (3.15) beruhen nicht nur auf der Tatsache, daß ADFT-Codes über GF(q) für jede Länge n mit (n,q) = 1 und (im Gegensatz zu zyklischen Codes) jede Dimension k mit $1 \leq k \leq n$ existieren.

(3.16) Beobachtung:

Es gibt Parameter q, n und d, für ein ADFT-Code größerer Dimension als jeder zyklischer Code mit diesen Parametern existiert.

(3.17) Beispiel:

Der binäre 1-fehler-korrigierende BCH-Code der Länge 9, der durch das Generatorpolynom x^6+x^3+1 definiert wird, besitzt Minimaldistanz 3 und Dimension 3.

Der binäre ADFT-Code der Länge 9, der durch die Kontrollfrequenzen Z = {3,4,5,6} definiert wird, besitzt Minimaldistanz 3 und Dimension 5.

Weitere Beispiele sind im Anhang (A3) angegeben. Das Vorhandensein (selbst unendlich vieler) "guter" Codes in einer Codeklasse reicht nicht hin, das Prädikat "gut" auch der Klasse zuzusprechen.

(3.18) Definition:

Eine Klasse linearer Codes über einem Alphabet GF(q) heißt gut, wenn sie eine unendliche Familie $C_1, C_2, \ldots$ von Codes mit den Parametern n_i, k_i und d_i enthält, so daß

$$\lim_{i\to\infty} k_i/n_i > 0 \quad \text{und} \quad \lim_{i\to\infty} d_i/n_i > 0 .$$

(3.19) Bemerkungen:

(a) Die Klasse der (obgleich perfekten) Hamming-Codes ist nach diesem Maßstab "schlecht". Auch die Klasse der primitiven BCH-Codes ist nicht gut (cf. /Li67/).

(b) Für die Klasse der zyklischen Codes ist nicht bekannt, ob sie gut ist (cf. Research Problem 9.2 in /Ma78/; siehe auch /Be74/).

(c) Gute Teilklassen der Klasse der linearen Codes sind die Alternant- und die Goppa-Codes (cf. /Ma78/).

(3.20) Beobachtung:

Es gibt eine unendliche Familie binärer ADFT-Codes, die die Griesmer-Schranke (siehe (I.2.3)) erreichen. Für $n \equiv 3 \bmod 6$ wird durch die Menge der Kontrollfrequenzen $Z = \lfloor 0:n-1 \rfloor \setminus \{n/3, 2n/3\}$ ein $(n,2)$-ADFT-Code mit Generatormatrix

$$G = \begin{bmatrix} 110\ 110\ 110\ \ldots \\ 101\ 101\ 101\ \ldots \end{bmatrix} .$$

definiert, der die Minimaldistanz 2n/3 besitzt. Wegen $N(2,2n/3,2) = n$ ist dieser Code optimal bezüglich der Griesmer-Schranke.

(3.21) Satz:

Zu jedem Paar (k,d) gibt es eine Minimaldistanz $d' \geq d$, für die ein bezüglich der Griesmer-Schranke optimaler binärer (n,k)-ADFT-Code existiert.

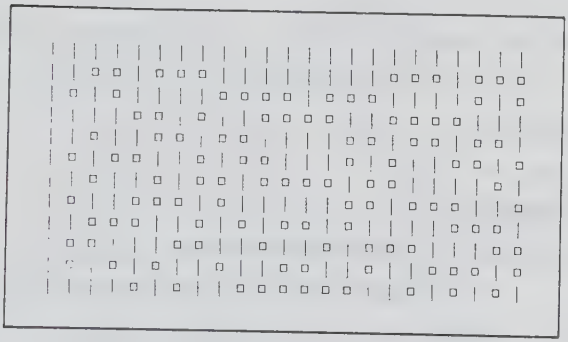
Beweis: Sei $a = d \text{ div } 2^{k-1}$ und $b = a + 1 - (a \bmod 2)$. Wähle $d' = b \cdot 2^{k-1}$. Mit Beobachtung (I.2.4) erhält man $N(k,d',2) \geq b \cdot (2^k-1)$. Der binäre $(b \cdot (2^k-1),k)$-ADFT-Code mit den Kontrollfrequenzen $Z = [0:n-1] \setminus \{b, 2 \cdot b, \ldots, k \cdot b\}$ besitzt die Minimaldistanz d'.

□

(3.22) Bemerkung:

Obige Codes entsprechen den in Beobachtung (II.3.3) konstruierten. Sie können als b-fache Wiederholung eines $(2^k-1,k)$-Simplex-Codes aufgefaßt werden. Abgesehen vom Nullvektor besitzen alle Codeworte eines $(2^r-1,r)$-Simplex-Codes dasselbe Gewicht 2^{r-1} (cf. /Ma78/).

Schließlich sei die Generatormatrix eines 3-fehlerkorrigierenden, perfekten (23,12)-ADFT-Codes über GF(2) (äquivalent zum (23,12)-Golay-Code) abgebildet. Sie spiegelt die vielfältigen Symmetrie-Eigenschaften dieses Codes weit deutlicher wider als die bekannten Generatormatrizen (vgl. z.B. die in /Ma78/ angegebenen).



§4 Zur Implementierung von ADFT-Codes

ADFT-Codes sind lineare Codes und können somit als solche codiert und decodiert werden. Verfahren zur Implementierung linearer Codes werden z.B. in /Be68/, /Pe72/, /Ma78/, /Bl83/ und /He83/ beschrieben. Die in diesem Abschnitt vorgestellten Möglichkeiten der Implementierung von ADFT-Codes basieren auf der Implementierung der ADFT (Decodierer) bzw. der IADFT (Codierer).

Die Codierung von ADFT-Codes kann analog zur Codierung zyklischer Codes mittels DFT (vgl. Kapitel III) mit Hilfe der ADFT durchgeführt werden.

(4.1) **Beobachtung:**

Sei C ein (n,k)-ADFT-Code über GF(q).

- Durch Nullsetzen der Kontrollfrequenzen von C und Besetzen der verbleibenden Koeffizienten eines Vektors $\underline{C} \in GF(q)^n$ mit Informationssymbolen wird ein Codewort im Spektralbereich erzeugt.
- Der aus der Anwendung der IADFT auf $\underline{C}$ resultierende Signalvektor $\underline{c}$ stellt das (nicht systematisch codierte) Codewort im Signalbereich dar.

Den Implementierungsmöglichkeiten der ADFT entsprechend erhält man serielle oder parallele Codierer. Im Vergleich zu den bekannten Codierverfahren für zyklische Codes (vgl. etwa /Pe72/, /Ma78/) ist die Codierung von ADFT-Codes i.a. zwar aufwendiger, bietet jedoch beachtenswerte Vorteile:

(4.2) Bemerkungen:

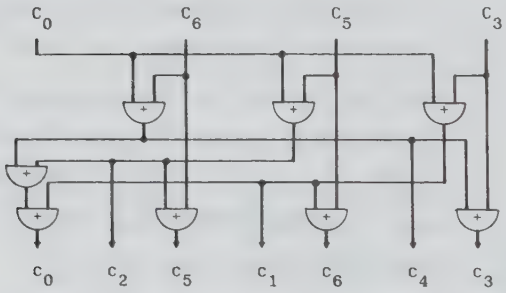
(a) Ein ADFT-Codierer für einen (n,k)-ADFT-Code über einem Körper GF(q) kann zur Codierung aller mit Hilfe derselben ADFT-Matrix definierbaren ADFT-Codes der Länge n über GF(q) eingesetzt werden.

(b) Durch die Anwendung von Faktorisierungsverfahren für die ADFT können (eventuell mit Abstrichen bei der Verarbeitungsgeschwindigkeit) Codierer für Codes relativ großer Länge mit bescheidenem Aufwand realisiert werden (siehe §2 und Anhang (A2)).

Verzichtet man auf den in Bemerkung (4.2a) genannten Vorteil, so kann die parallele Implementierung eines ADFT-Codierers durch Berücksichtigung der Kontrollfrequenzen des Codes vereinfacht werden.

(4.3) Beispiel:

Der binäre (7,4)-ADFT-Code aus Beispiel (3.4) besitzt die Kontrollfrequenzen (1,2,4). Zur parallelen Realisierung eines Codierers für diesen Code (Abbildung (4.4)) sind 8 XOR-Gatter ausreichend (die Implementierung der IADFT erforderte 14 XOR-Gatter; vgl. Beispiel (2.4) und Abbildung (2.5)).



(4.4) Abbildung: Parallele Codierung eines (7,4)-ADFT-Codes

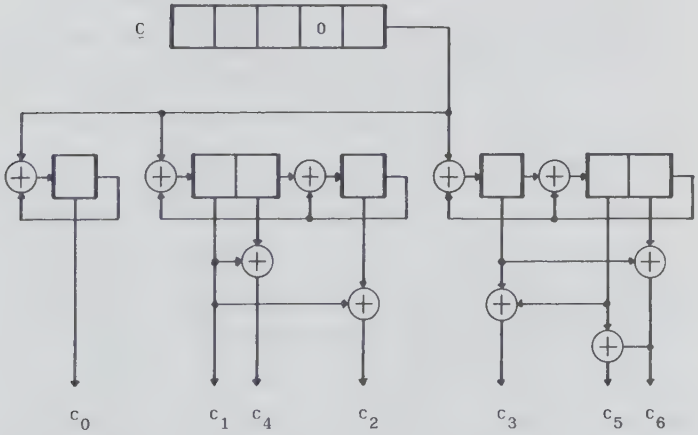
Durch die geschickte Wahl der Kontrollfrequenzen eines ADFT-Codes kann auch die serielle Codierung vereinfacht werden.

(4.5) Beobachtung:

Ist {n-1,n-2,...,n-i} Teilmenge der Menge der Kontrollfrequenzen eines (n,k)-ADFT-Codes, so sind zu dessen serieller Codierung n-i Takte ausreichend.

(4.6) Beispiel:

Sei q = 2, n = 7, A_7 die 7-te ADFT-Matrix aus Beispiel (1.4) und C der durch Z = {3,5,6} definierte (7,4)-ADFT-Code. C ist nach Satz (3.10) äquivalent zum (7,4)-ADFT-Code mit den Kontrollfrequenzen {1,2,4} und kann in 5 Takten mit der in Abbildung (4.7) gezeigten Schaltung seriell codiert werden.



(4.7) Abbildung: Serielle Codierung eines (7,4)-ADFT-Codes

Zur Decodierung eines linearen Blockcodes wird bei den gängigen Decodierverfahren (cf. /Ma78/, /Bl83/) zunächst das Syndrom des empfangenen Wortes berechnet.

(4.8) Definition:

Sei C ein linearer (n,k)-Code über $GF(q)$ und H eine Kontrollmatrix von C. Das **Syndrom** s eines Vektors $\underline{y} \in GF(q)^n$ ist gegeben durch

$$\underline{s} = \underline{y} \cdot H^t .$$

(4.9) Bemerkung:

Ist C ein t-fehlerkorrigierender (n,k)-Code über $GF(q)$, so ist die Menge aller $\underline{y} \in GF(q)^n$, die dasselbe Syndrom besitzen, eine Nebenklasse von C und enthält höchstens einen (falls C perfekt ist: genau einen) Vektor $\underline{e}$ mit $wgt(\underline{e}) \leq t$.

(4.10) Beobachtung:

Sei C ein durch eine n-te ADFT-Matrix ${}^{B}A_n$ über $GF(q)$ und die Menge der Kontrollfrequenzen $Z = \{z[0], z[1], \ldots, z[n-k-1]\}$ definierter (n,k)-ADFT-Code. Dann ist für $\underline{y} \in GF(q)^n$ und $\underline{Y} = \underline{y} \cdot {}^{B}A_n$ das Syndrom von $\underline{y}$ gegeben durch

$$\underline{s} = (Y_{z[0]}, Y_{z[1]}, \ldots, Y_{z[n-k-1]}) .$$

Beobachtung (4.10) lehrt, daß für die Klasse der ADFT-Codes die Codierung und die Syndromberechnung im wesentlichen mit derselben Hardware durchgeführt werden kann.

In dem als *'coset-leader'-Algorithmus* bekannten Decodierverfahren für lineare Blockcodes (cf. /Bl83/) wird die in Bemerkung (4.9) angegebene Eigenschaft des Syndroms dadurch ausgenutzt, daß zur Bestimmung des aufgetretenen Fehlers eine Tabelle von Nebenklassenvertretern mit minimalem Gewicht gespeichert (und durch das Syndrom adressiert) wird.

(4.11) Beobachtung:

Die zur Decodierung eines nicht systematisch codierten (n,k)-Codes C über $GF(2^m)$ mit dem 'coset-leader'-Algorithmus nötige Tabelle der Nebenklassenvertreter umfaßt $2^{m(n-k)}$ Einträge a m·n Bit. Bei systematischer Codierung kommt man mit einem $m \cdot k \cdot 2^{m(n-k)}$-Bit-Speicher aus.

Die Implementierung dieses Decodierverfahrens scheitert für viele praxisrelevante Codes am Speicherplatzbedarf. Die Speicherkapazität eines heute erhältlichen ROM-Bausteins (2^{18} Bit) ist z.B. für die 'coset-leader'-Tabelle des binären (511,502)-Hamming-Codes ausreichend. Zur 'coset-leader'-Decodierung des 3-fehlerkorrigierenden (31,25)-Reed-Solomon-Codes über $GF(2^5)$ dagegen benötigte man 20480 derartige Speicherbausteine.

Zur Reduktion des Tabellenumfangs können die Automorphismen eines Codes ausgenutzt werden.

(4.12) Beobachtung:

Sei C ein t-fehlerkorrigierender (n,k)-Code über GF(q) und G eine Gruppe von Automorphismen von C. G induziert eine Äquivalenzrelation auf der Menge der Vektoren $\underline{e} \in GF(q)^n$ mit $wgt(\underline{e}) \leq t$.

Beobachtung (4.12) kann in naheliegender Weise zur Implementierung eines *Permutations-Decodierverfahrens* (cf. /Ma78/) genutzt werden. Für dieses Verfahren ist nur die Speicherung eines Repräsentanten jeder Äquivalenzklasse korrigierbarer Fehlermuster erforderlich. Es kann immer dann besonders günstig implementiert werden, wenn für jedes σ aus der Gruppe G von Automorphismen von C das Syndrom von $\sigma(\underline{y})$ mit geringem Aufwand aus dem Syndrom von $\underline{y}$ bestimmt werden kann.

(4.13) Satz:

Sei C ein (n,k)-ADFT-Code über GF(q) und

$$\sigma : i \dashrightarrow i+1 \bmod n$$

Automorphismus von C. Dann kann für $\underline{y} \in GF(q)^n$ das Syndrom von $\sigma(\underline{y})$ bei seriell implementierter ADFT in einem Takt aus dem Syndrom von $\underline{y}$ berechnet werden.

Beweis: Sei α n-te Einheitswurzel über GF(q) und M(x) das Minimalpolynom von α. Bei serieller Implementierung der ADFT führt das zu M(x) gehörige Rückkopplungs-Schieberegister bei Input 0 in einem Takt eine Multiplikation mit α durch (/cf. Be68/) und die Behauptung folgt aus der Phaseneigenschaft der ADFT (siehe (1.10)).

□

(4.14) Satz:

Sei C ein (n,k)-ADFT-Code über GF(q) und

$$\sigma : i \dashrightarrow t \cdot i \bmod n$$

Automorphismus von C. Dann ist für $\underline{y} \in GF(q)^n$ das Syndrom von $\sigma(\underline{y})$ eine Permutation des Syndroms von $\underline{y}$.

Beweis: Die Behauptung folgt unmittelbar aus der Skalierungseigenschaft der ADFT (siehe (1.11)).

□

(4.15) Bemerkung:

Der *Meggitt-Decodierer* ist ein Permutations-Decodierverfahren das nur die Invarianz zyklischer Codes gegenüber zyklischem Schieben zur Reduktion der Tabelle der Restklassenvertreter nutzt (cf. /Bl83/). Es vermindert den Speicherplatzbedarf auf $m \cdot 2^{m(n-k)}$ Bit.

Anhang

(A1) Normalbasen von $GF(2^m)$ über GF(2)

Im folgenden sind für $2 \leq m \leq 12$ (nach Möglichkeit selbst-duale) Normalbasen von $GF(2^m)$ über GF(2) durch die zugehörigen Kreisteilungsklassen modulo $n = 2^m-1$ angegeben. M(x) bezeichnet dabei das zur Körpererweiterung benutzte primitive Polynom; "NB" steht für Normalbasis.

m = 2 n = 3 M(x) = x^2+x+1	NB	dual zu
K(0,2) = { 0 }		
K(1,2) = { 1,2 }	ja	selbst-dual

m = 3 n = 7 M(x) = x^3+x+1	NB	dual zu
K(0,2) = { 0 }		
K(1,2) = { 1,2,4 }	nein	
K(3,2) = { 3,6,5 }	ja	selbst-dual

m = 4 n = 15 M(x) = x^4+x+1	NB	dual zu
K(0,2) = { 0 }		
K(1,2) = { 1,2,4,8 }	nein	
K(3,2) = { 3,6,12,9 }	ja	K(7,2)
K(5,2) = { 5,10 }		
K(7,2) = { 7,14,13,11 }	ja	K(3,2)

m = 5 n = 31 $M(x) = x^5+x^2+1$	NB	dual zu
K(0,2) = { 0 }		
K(1,2) = { 1,2,4,8,16 }	nein	
K(3,2) = { 3,6,12,24,17 }	ja	K(11,2)
K(5,2) = { 5,10,20,9,18 }	ja	selbst-dual
K(7,2) = { 7,14,28,25,19 }	nein	
K(11,2) = { 11,22,13,26,21 }	ja	K(3,2)
K(15,2) = { 15,30,29,27,23 }	nein	

m = 6 n = 63 $M(x) = x^6+x+1$	NB	dual zu
K(0,2) = { 0 }		
K(1,2) = { 1,2,4,8,16,32 }	nein	
K(3,2) = { 3,6,12,24,48,33 }	nein	
K(5,2) = { 5,10,20,40,17,34 }	ja	K(31,2)
K(7,2) = { 7,14,28,56,49,35 }	nein	
K(9,2) = { 9,18,36 }		
K(11,2) = { 11,22,44,25,50,37 }	nein	
K(13,2) = { 13,26,52,41,19,38 }	nein	
K(15,2) = { 15,30,60,57,51,39 }	ja	selbst-dual
K(21,2) = { 21,42 }		
K(23,2) = { 23,46,29,58,53,43 }	ja	selbst-dual
K(27,2) = { 27,54,45 }		
K(31,2) = { 31,62,61,59,55,47 }	ja	K(5,2)

m = 7 n = 127 $M(x) = x^7+x+1$	NB	dual zu
K(13,2)	ja	K(31,2)
K(19,2)	ja	K(63,2)
K(21,2)	ja	K(27,2)
K(43,2)	ja	selbst-dual

m = 8 n = 255 M(x) = $x^8+x^6+x^5+x^4+1$	NB	dual zu
K(3,2)	ja	K(95,2)
K(5,2)	ja	K(111,2)
K(11,2)	ja	K(27,2)
K(13,2)	ja	K(23,2)
K(15,2)	ja	K(61,2)
K(21,2)	ja	K(43,2)
K(25,2)	ja	K(87,2)
K(37,2)	ja	K(53,2)

m = 9 n = 511 M(x) x^9+x^4+1	NB	dual zu
K(41,2)	ja	selbst-dual
K(77,2)	ja	selbst-dual
K(117,2)	ja	selbst-dual
und 18 nicht selbst-duale Normalbasen		

m = 10 n = 1023 M(x) = $x^{10}+x^3+1$	NB	dual zu
K(17,2)	ja	selbst-dual
K(55,2)	ja	selbst-dual
K(59,2)	ja	selbst-dual
K(221,2)	ja	selbst-dual
und 44 nicht selbst-duale Normalbasen		

m = 11 n = 2047 M(x) = $x^{11}+x^2+1$	NB	dual zu
K(121,2)	ja	selbst-dual
K(331,2)	ja	selbst-dual
K(439,2)	ja	selbst-dual
und 90 nicht selbst duale Normalbasen		

m = 12 n = 4095	128 nicht selbst-duale Normalbasen

(A2) Aufwand für die parallele Implementierung der ADFT

In nebenstehender Tabelle sind für Längen $3 \leq n \leq 3255$ die Zahlen der zur parallelen Implementierung einer binären ADFT nötigen Bitoperationen

(A1) bei direkter paralleler Berechnung der ADFT

(A2) bei paralleler Berechnung der ADFT mit einer aus einem Minimisierungsverfahren (ein einfacher 'greedy'-Algorithmus) resultierenden Schaltung

(A3) bei paralleler Berechnung der ADFT mit Hilfe paralleler Faktorisierung (siehe Kapitel IV.2)

und zum Vergleich (cf. /Sa78/)

(D1) bei direkter paralleler Berechnung der DFT

(D2) bei "schneller" paralleler Berechnung der DFT

gegenübergestellt. In der Tabelle ist ferner der Erweiterungsgrad m des Zerfällungskörpers von x^n-1 über GF(2), sowie die Inversionseigenschaft (IE) der jeweiligen ADFT-Matrix angegeben.

n	m	IE	A1	A2	A3	D1	D2
3	2	ja	4	3			
5	4		16	9			
7	3	ja	24	14		216	126
9	6	ja	48	28			
11	10	ja	60	25			
13	12	-	72	40			
15	4		132	57	58	1788	594
17	8	-	160	82			
21	6	ja	244	114	63		
23	11	ja	264	72			
31	5	ja		< 343		12000	2790
35	12				133		
63	6	ja		< 1240		71412	8902
69	22	ja			285		
105	12	-			609		
119	24	-			812		
127	7	ja		< 4411		395136	39438
253	110	ja			1367		
255	8	-		< 15939		2069788	152950
483	66	-			2961		
511	9	ja		< 57404		$> 10^7$	530994
1023	10	ja		< 210000		$> 5 \cdot 10^7$	1948238
1771	330	ja			13111		
2047	11	ja		< 800000		$> 2 \cdot 10^8$	8338585
3255	60	-			< 54600		

(A3) Binäre ADFT-Codes für $n \leq 23$

Für den Bereich $3 \leq n \leq 23$ und $2 \leq k < n$ wurden mit Hilfe eines Mikrocomputers die Minimaldistanzen aller binären (n,k)-ADFT-Codes ermittelt. Ein kleiner Teil der Resultate ist in der untenstehenden Tabelle angegeben. Für jedes Paar (n,k) ist ein (wenn möglich zyklischer) (n,k)-ADFT-Code mit der Minimaldistanz $d = d_{max}(n,k)$ aufgeführt, falls $d > 2$ gilt. Für verschiedene Parameter ist zum Vergleich ein zyklischer (n,k)-Code mit maximaler Minimaldistanz mit aufgelistet. Die Kontrollfrequenzen sind nach Kreisteilungsklassen geordnet angegeben. Die Kreisteilungsklassen sind dabei durch '.' getrennt; "0.110.000" entspricht z.B. der Menge der Kontrollfrequenzen $Z = \{0,4,3,6,5\}$.

n	k	d	Kontrollfrequenzen	Bem.
7	2	4	0.110.000	
7	3	4	0.111.000	zykl.
7	4	3	1.111.000	zykl.
9	2	6	0.000000.11	zykl.
(9	3	3	1.000000.11	zykl.)
9	3	4	0.111000.00	
9	4	4	0.110110.00	
9	5	3	1.110110.00	
11	2	6	0.1100000000	
11	3	4	0.1110000000	
11	4	4	0.1111000000	
11	5	4	0.1111100000	
11	6	3	1.1111100000	

n	k	d	Kontrollfrequenzen	Bem.
13	2	6	1.100000000000	
13	3	6	1.110000000000	
13	4	6	0.111100000000	
13	5	4	0.111010010000	
13	6	4	0.111111000000	
13	7	4	0.111110010100	
13	8	3	1.111110010100	
15	2	10	0.0000.0000.11.0000	zykl.
15	3	5	1.0000.0000.11.0000	zykl.
15	3	8	0.1100.0001.00.0000	
15	4	8	0.1111.0000.00.0000	zykl.
15	5	7	1.1111.0000.00.0000	zykl.
15	6	6	0.0000.1111.11.0000	zykl.
15	7	5	1.1111.0000.11.0000	zykl.
15	8	4	1.0000.1111.00.1111	zykl.
15	9	4	1.1111.0000.00.1111	zykl.
15	10	4	0.1111.1111.11.0000	zykl.
15	11	3	1.1111.1111.11.0000	zykl.
17	2	10	0.00000001.00000001	
17	3	8	0.00000001.10000010	
17	4	8	0.00000101.00001010	
17	5	6	1.00000011.01010000	
17	6	6	0.00000101.11101000	
17	7	6	0.00001001.10011110	
17	8	6	0.00000000.11111111	zykl.
17	9	5	1.00000000.11111111	zykl.
17	10	4	0.00100111.10110111	
17	11	3	1.00110011.00111111	

n	k	d	Kontrollfrequenzen	Bem.
21	2	14	0.000000.000.000000.11.000	zykl.
21	3	12	0.000000.111.000000.00.000	zykl.
(21	4	9	1.000000.000.000000.00.111	zykl.)
21	4	10	0.000000.111.000000.10.000	
21	5	10	0.000000.111.000000.11.000	zykl.
21	6	8	0.000000.000.111111.00.000	zykl.
21	7	8	1.111111.000.000000.00.000	zykl.
(21	8	6	0.000000.111.000000.11.111	zykl.)
21	8	8	0.111111.110.000000.00.000	
21	9	8	0.111111.111.000000.00.000	zykl.
(21	10	5	1.000000.000.111111.00.111	zykl.)
21	10	6	0.111111.111.000000.10.000	
21	11	6	0.111111.111.000000.11.000	zykl.
21	12	5	1.111111.111.000000.11.000	zykl.
21	13	4	1.111111.000.111111.00.000	zykl.
21	14	4	0.111111.111.000000.11.111	zykl.
21	15	4	0.111111.111.111111.00.000	zykl.
21	16	3	1.111111.111.111111.00.000	zykl.
23	2	12	0.11000000000.00000000000	
23	3	12	0.11100000000.00000000000	
23	4	11	1.10101000000.00000000000	
23	5	8	1.10101000100.00000000000	
23	6	8	0.11110100100.00000000000	
23	7	8	0.11111100100.00000000000	
23	8	8	0.11111110100.00000000000	
23	9	8	0.11111111100.00000000000	
23	10	8	0.11111111110.00000000000	
23	11	8	0.11111111111.00000000000	zykl.
23	12	7	1.11111111111.00000000000	zykl.

Literatur

/Ah75/ Ahmed, N.; Rao, K.R.:
'Orthogonal Transforms for Digital Signal Processing',
(Springer, Berlin, 1975)

/Ba63/ Bartee, T.C.; Schneider, D.I.:
'Computation with Finite Fields',
Information and Control, **6** (1963), 79-98

/Ba73/ Baumert, L.D.; McEliece, R.J.:
'A Note on the Griesmer Bound',
IEEE Trans. Information Theory, **19** (1973), 134-135

/Be74/ Beard, J.T.B.:
'Computing in GF(q)',
Mathematics of Computation, **28** (1974), 1159-1168

/Be68/ Berlekamp, E.R.:
'Algebraic Coding Theory',
(McGraw-Hill, New York, 1968)

/Be82a/ Berlekamp, E.R.:
'Bit-Serial Reed-Solomon Encoders',
IEEE Trans. Information Theory, **28** (1982), 869-874

/Be74/ Berlekamp, E.R.; Justesen, J.:
'Some Long Cyclic Linear Binary Codes Are Not So Bad',
IEEE Trans. Information Theory, **20** (1974), 351-356

/Be80/ Beth, T.:
'Codes in Digital Signal Processing',
CISM Lecture Notes, 1980

/Be82b/ Beth, T.:
'Some Aspects of Coding Theory between Probability, Algebra, Combinatorics and Complexity Theory',
in: Jungnickel, D.; Vedder, K. (eds):
'Combinatorial Theory',
Lecture Notes in Math. **969**, Springer, Berlin, 1982

/Be84a/ Beth, T.:
'Die Algebraische Diskrete Fourier-Transformation',
(Abschlußbericht zum DFG-Projekt Be/887-1, 1984)

/Be84b/ Beth, T.:
'Verfahren der schnellen Fourier-Transformation',
(Teubner, Stuttgart, 1984)

/Be83a/ Beth, T.; Fumy, W.:
'Hardware-Oriented Algorithms for the Fast Symbolic Calculation of the DFT',
Electronics Letters, **19** (1983), 901-902

/Be83b/ Beth, T.; Fumy, W.; Mühlfeld, R.:
'Zur Algebraischen Diskreten Fourier-Transformation',
Arch. Math., **40** (1983), 238-244

/Be85/ Beth, T.; Jungnickel, D.; Lenz, H.:
'Design Theory',
(Bibliographisches Institut, Mannheim, 1985)

/Bl79/ Blahut, R.E.:
'Transform Techniques for Error Control Codes',
IBM J. Res. Develop., **23** (1979), 299-315

/Bl82/ Blahut, R.E.:
'Fast Algorithms for Reed-Solomon Decoders',
CISM Lecture Notes, 1982

/Bl83/ Blahut, R.E.:
'Theory and Practice of Error Control Codes',
(Addison-Wesley, Reading, Mass., 1983)

/Bl84/ Blahut, R.E.:
'A Universal Reed-Solomon Decoder',
IBM J. Res. Develop., **28** (1984), 150-158

/Bo60/ Bose, R.C.; Ray-Chaudhuri, D.K.:
'On a Class of Error Correcting Binary Group Codes',
Information and Control, **3** (1960), 68-79

/Ca82/ Carasso, M.G.; Peek, J.B.H.; Sinjou, J.P.:
'The Compact Disc Digital Audio System',
Philips Tech. Rev., **40** (1982), 151-155

/Ch70/ Chen, C.L.:
'Computer Results on the Minimum Distance of Some
Binary Cyclic Codes',
IEEE Trans. Information Theory, **16** (1970), 359-360

/Ch69/ Chen, C.L.; Peterson, W.W.; Weldon, E.J.:
'Some Results on Quasi-Cyclic Codes',
Information and Control, **15** (1969), 407-423

/Ch75/ Chien, R.T.; Choy, D.M.:
'Algebraic Generalisation of BCH-Goppa-Helgert Codes',
IEEE Trans. Information Theory, **21** (1975), 70-79

/Cl81/ Clark, G.C.; Cain, J.B.:
'Error-Correction Coding for Digital Communications',
(Plenum Press, New York, 1981)

/Co68/ Conway, J.H.:
'A Tabulation of Some Information Concerning Finite Fields'
in: Churchhouse/Herz (eds.):
'Computers in Mathematical Research',
(North-Holland, Amsterdam, 1968)

/Da68/ Davenport, H.:
'Bases for Finite Fields',
J. London Math. Soc., **43** (1968), 21-39

/De75/ Delsarte, P.:
'On Subfield Subcodes of Modified Reed-Solomon Codes',
IEEE Trans. Information Theory, **21** (1975), 575-576

/El55/ Elias, P.:
'Coding for Noisy Channels',
IRE Convention Record, Part 4, (1955), 37-45

/Fo65/ Forney, G.D.:
'On Decoding BCH-Codes',
IEEE Trans. Information Theory, **11** (1965), 549-557

/Fu84/ Fumy, W.; Posch, S.; Rieß, H.P.:
'Materialien zur Codierungstheorie III:
Bündelfehler-korrigierende Codes und ihre Implementierung
für die Magnetbandaufzeichnung',
Arbeitsberichte des IMMD, Erlangen, 17/7 (1984)

/Go67/ Golomb, S.W.:
'Shift Register Sequences',
(Holden-Day, San Francisco, 1967)

/Go58/ Good, I.J.:
'The Interaction Algorithm and Practical Fourier Analysis',
J. Royal Stat. Soc., **20** (1958), 361-375

/Go70/ Goppa, V.D.:
'A New Class of Linear Error-Correcting Codes',
Problems of Info. Transmission, **6** (1970), 207-212

/Go82/ Gordon, D.M.:
'Minimal Permutation Sets for Decoding the Binary Golay Codes',
IEEE Trans. Information Theory, **28** (1982), 541-543

/Gr60/ Griesmer, J.H.:
'A Bound for Error-Correcting Codes',
IBM J. Res. Develop., **4** (1960), 532-542

/Ha72a/ Hartmann, C.R.P.; Tzeng, K.K.:
'Generalizations of the BCH Bound',
Information and Control, **20** (1972), 489-498

/Ha72b/ Hartmann, C.R.P.:
'Decoding Beyond the BCH Bound',
IEEE Trans. Information Theory, **18** (1972), 441-444

/Ha74/ Hartmann, C.R.P.; Tzeng, K.K.:
'Decoding Beyond the BCH Bound Using Multiple Sets of Syndrome Sequences',
IEEE Trans. Information Theory, **20** (1974), 292-295

/He83/ Heise, W.; Quattrocchi, P.:
'Informations- und Codierungstheorie',
(Springer, Berlin, 1983)

/He72/ Helgert, H.J.:
'Noncyclic Generalizations of BCH and Srivastava codes',
Information and Control, **21** (1972), 280-290

/He73/ Helgert, H.J.; Stinaff, R.D.:
'Minimum-Distance Bounds for Binary Linear Codes',
IEEE Trans. Information Theory, **19** (1973), 344-356

/Ho59/ Hocquenghem, A.:
'Codes Correcteurs d'Erreurs',
Chiffres, **2** (1959), 147-156

/Ho82/ Hoeve, H.; Timmermans, J.; Vries, L.B.:
'Error Correction and Concealment in the Compact Disc System',
Philips Tech. Rev., **40** (1982), 166-172

/Hs84/ Hsu, I.; Reed, I.S.; Truong, T.K.; Wang, K.; Yeh, C.; Deutsch, L.J:
'The VLSI Implementation of a Reed-Solomon Encoder Using Berlekamp's Bit-Serial Multiplier Algorithm',
IEEE Trans. Computers, **33** (1984), 906-911

/Ke78/ Kernighan, B.W.; Ritchie, D.M.:
'The C Programming Language',
(Prentice-Hall, London, 1978)

/La71/ Laws, B.A.; Rushforth, C.K.:
'A Cellular-Array Multiplier for GF(2^m)',
IEEE Trans. Computers, **20** (1971), 1573-1578

/Le75/ Lempel, A.:
'Matrix Factorization over GF(2) and Trace-Orthogonal Bases of GF(2^m)',
SIAM J. Comput., **4** (1975), 175-186

/Le77/ Lempel, A.; Winograd, S.:
'A New Approach to Error-Correcting Codes',
IEEE Trans. Information Theory, **23** (1977), 503-508

/Li83/ Lidl, R.; Niederreiter, H.:
'Finite Fields',
(Addison-Wesley, Reading, Mass., 1983)

/Li84/ Lidl, R.; Pilz, G.:
'Applied Abstract Algebra',
(Springer, New York, 1984)

/Li67/ Lin, S.; Weldon, E.J.:
'Long BCH Codes Are Bad',
Information and Control, 11 (1967), 445-451

/Li81/ Lipson, J.D.:
'Elements of Algebra and Algebraic Computing',
(Addison-Wesley, Reading, Mass., 1981)

/Li84/ Liu, K.Y.:
'Architecture for VLSI Design of Reed-Solomon Decoders',
IEEE Trans. Computers, 33 (1984), 178-189

/Lü79/ Lüneburg, H.:
'Galoisfelder, Kreisteilungskörper und
Schieberegisterfolgen',
(Bibliographisches Institut, Mannheim, 1979)

/Ma60/ MacDonald, J.E.:
'Design Methods for Maximum Minimum-Distance Error-
Correcting Codes'
IBM J. Res. Develop., 4 (1960), 43-57

/Ma64/ MacWilliams, F.J.:
'Permutation Decoding of Systematic Codes',
Bell System Technical Journal, 43 (1964), 134-138

/Ma78/ MacWilliams, F.J.; Sloane, N.J.A.:
'The Theory of Error-Correcting Codes',
(North-Holland, Amsterdam, 1978)

/Ma84/ Mandelbaum, D.M.:
'Reducing the Number of Operations in Certain Finite-Field Transforms',
IEEE Trans. Information Theory, **30** (1984), 546-547

/Ma69/ Massey, J.L.:
'Shift-Register Synthesis and BCH-Decoding',
IEEE Trans. Information Theory, **15** (1969), 122-127

/Ma61/ Mattson, H.F.; Solomon, G.:
'A New Treatment of Bose-Chaudhuri Codes',
J. Soc. Indust. Appl. Math., **9** (1961), 654-669

/Mi79/ Miller, R.L.:
'Generalized BCH Codes'
Information and Control, **40** (1979), 61-75

/Mi62/ Mitchell, M.E.:
'Simple Decoders and Correlators for Cyclic Error-Correcting Codes'
IRE Trans. Communication Systems, **10** (1962), 284-290

/Mü82/ Mühlfeld, R.:
'Methoden der Algebraischen Diskreten Fourier-Transformation',
Diplomarbeit, Universität Erlangen, 1982

/Nu81/ Nussbaumer, H.J.:
'Fast Fourier Transform and Convolution Algorithms',
(Springer, Berlin, 1981)

/Pe60/ Peterson, W.W.:
'Encoding and Error-Correction Procedures for the Bose-Chaudhuri-Codes',
IRE Trans. Information Theory, 6 (1960), 459-470

/Pe72/ Peterson, W.W.; Weldon, E.J.:
'Error Correcting Codes', 2nd ed.
(MIT Press, Cambridge, Mass., 1972)

/Po71/ Pollard, J.M.:
'The Fast Fourier Transform in a Finite Field',
Mathematics of Computation, 25 (1971), 365-374

/Po84/ Posch, S.:
'Pilotstudie zur Algebraischen Diskreten Fourier-Transformation',
Studienarbeit, Universität Erlangen, 1984

/Pr62/ Prange, E.:
'The Use of Information Sets in Decoding Cyclic Codes'
IEEE Trans. Information Theory, 8 (1962), S5-S9

/Pr77/ Preparata, F.P.; Sarwate, D.V.:
'Computational Complexity of Fourier Transforms over Finite Fields',
Mathematics of Computation, 31 (1977), 740-751

/Re60/ Reed, I.S.; Solomon, G.:
'Polynomial Codes over Certain Finite Fields',
J. Soc. Indust. Appl. Math., 8 (1960), 300-304

/Ro82/ Roos, C.:
'A Generalization of the BCH Bound for Cyclic Codes, Including the Hartmann-Tzeng Bound',
J. Combinatorial Theory (A), 33 (1982), 229-232

Ro83/ Roos, C.:
'A New Lower Bound for the Minimum Distance of a Cyclic Code',
IEEE Trans. Information Theory, 29 (1983), 330-332

Ru69/ Rushforth, C.K.:
'Fast Fourier-Hadamard Decoding of Orthogonal Codes',
Information and Control, 15 (1969), 33-37

Sa78/ Sarwate, D.V.:
'Semi-Fast Fourier Transforms over GF(2^m)',
IEEE Trans. Computers, 27 (1978), 283-285

Se66/ Selmer, E.S.:
'Linear Recurrence Relations over Finite Fields',
(Dept. of Math., University of Bergen, Norway, 1966)

Sh48/ Shannon, C.:
'A Mathematical Theory of Communication',
Bell System Technical Journal, 27 (1948), 379-423, 623 656

So65/ Solomon, G.; Stiffler, J.J.:
'Algebraically Punctured Cyclic Codes',
Information and Control, 8 (1965), 170-179

Ta68/ Tanaka, H.; Kasahara, M.; Tezuka, Y.; Kasahara, Y.:
'Computation over Galois Fields Using Shiftregisters',
Information and Control, 13 (1968), 75 84

Th63/ Thomas, L.H.:
'Using a Computer to Solve Problems in Physics',
in: Ginn (ed):
'Applications of Digital Computers',
(Boston, 1963)

/Th83/ Thompson, T.M.:
'From Error-Correcting Codes through Sphere Packings
to Simple Groups',
(Math. Assoc. of America, 1983)

/Wi80/ Willett, M.:
'Arithmetic in a Finite Field',
Mathematics of Computation, **35** (1980), 1353-1359

/Wo83/ Wolfmann, J.:
'A Permutation Decoding of the (24,12,8) Golay Code',
IEEE Trans. Information Theory, **29** (1983), 748-750

/Ye84/ Yeh, C.S.;Reed, I.S.; Truong, T.K.:
'Systolic Multipliers for Finite Fields GF(2^m)',
IEEE Trans. Computers, **33** (1984), 357-360

/Zi70/ Zierler, N.:
'On x^n+x+1 over GF(2)',
Information and Control, **16** (1970), 502-505

/Zi83/ Zimmermann, K.H.:
'Zur Konstruktion von Normalbasen über endlichen Körpern',
Studienarbeit, Universität Erlangen, 1983

Index

ADFT 94
  primitive - 94
Algorithmus
  Berlekamp- 91
  Berlekamp-Massey- 90
  'coset-leader'- 135
  Euklid'scher - 53
  Forney- 90
  Good-Thomas- 110

Bahn 18
Basis 30
  duale - 34
  Normal- 30
  orthogonale - 34
  polynomiale - 30
  schwach spurduale - 34

chinesischer Restesatz 110
Code 2
  ADFT- 116
    primitiver - 116
  äquivalente -s 5
  Alternant- 28
  -Automorphismus 16
  BCH- 25
    primitiver - 25
    verallgemeinerter - 28
  Block- 2
  dualer - 4
  fehlererkennender - 3
  fehlerkorrigierender - 3
  Golay- 129
  Goppa- 28
  Hamming- 26
  linearer - 4
  MDS- 8
  optimaler - 8
  perfekter - 10
  Reed-Muller- 59
  Reed-Solomon- 26
    verallgemeinerter - 27
  -Schrumpfung 23
  selbst-dualer - 4
  Simplex- 66
  Spur- 23
  systematischer - 5
  zyklischer - 12
Codewort 2
  - im Frequenzbereich 74,117
Codierung 2
  ADFT- 130
  DFT- 79
  systematische - 5

$d_{max}(n,k)$ 123
Decodierung 2
  Meggitt- 137
  Permutations- 136
DFT 70

Generator-
  -matrix 4
  -polynom 13

Hamming-
  -Abstand 3
  -Gewicht 3

Kanal 2
konjugiert 18
konsekutiv 21
Kontroll-
  -frequenz 74,116
  -matrix 4
  -polynom 15
Kreisteilungsklasse 18

Matrix
  ADFT- 94
    - mit Inversionseig. 97
  Begleit- 33
  Fourier- 70
  Generator- 4
  Hadamard- 60
    binäre - 61
    normalisierte - 61
  Kontroll- 4
Minimal-
  -distanz 3
    maximale - 123
  -gewicht 6
  -polynom 18
modulare Repräsentation 65

N(k,d,q) 9
Nachricht 2
Nullstelle 17

Polynom
  charakteristisches - 32
  Fehlerstellen- 89
  Fehlerwerte- 90
  Generator- 13
  Kontroll- 15
  Mattson-Solomon- 69
  Minimal- 18

Rate 4
Redundanz 4

Schranke 8
  BCH- 21
  Gilbert-Varshamov- 11
  Griesmer- 9
  Hamming- 10
  Plotkin- 11
  Singleton- 8
Spur 31
Syndrom 134

Transformation
  Diskrete Fourier- 70
    Algebraische - 94
  Walsh-Hadamard- 62

WHT 62

Lebenslauf

Walter Fumy,
geb. am 01.10.1953 in Nürnberg als ältester Sohn des Finanzbeamten Karl Fumy und seiner Ehefrau Emma Fumy, geb. Strobl;
verheiratet mit Christina Strobel-Fumy, Sprachlehrerin;
wohnhaft Kirschenstr. 10, 8524 Kleinsendelbach.

09/59-07/60	erstes Grundschuljahr an der Volksschule Hilpoltstein
09/60-07/63	Grundschulklassen 2 bis 4 an der Bismarckschule Nürnberg
09/63-06/72	Besuch des Hans-Sachs-Gymnasiums Nürnberg (mathematisch-naturwissenschaftlicher Zweig)
23.06.1972	Reifeprüfung
07/72-11/73	Wehr- und Zivildienst
11/73-10/80	Studium an der Friedrich-Alexander-Universität Erlangen/Nürnberg WS 73/74 Hauptfach: Mathematik, Nebenfach: Informatik ab SS 74 Hauptfach: Informatik, Nebenfach: Physiologie
29.04.1976	Informatik-Vordiplom
15.10.1980	Informatik-Hauptdiplom
02/75-01/80	freier Mitarbeiter der ELDICON-GmbH, Nürnberg; Bearbeitung verschiedener Software-Projekte
02/80-05/81	Leiter der Software-Abteilung o.g. Firma; Entwicklung von Betriebs- und Anwendungssoftware für Mikrocomputer
06/81-10/83	wiss. Mitarbeiter am DFG-Projekt "Algebraische Diskrete Fourier-Transformation" (Dr.Th.Beth, IMMD I)
seit 11/83	wiss. Mitarbeiter am IMMD I (Prof.Dr.K.Leeb)

In letzter Zeit sind folgende Arbeitsberichte des Instituts für Mathematische Maschinen und Datenverarbeitung erschienen:

Band 14

Nr. 1 Pattern Recognition
(Research at Lehrstuhl für Informatik 5)
(Februar 1981)

Nr. 2 Bunke, H.:
Analyse elektrischer Schaltpläne
(März 1981)

Nr. 3 Linster, C.-U.:
SYMPOS/UNIX - Ein Betriebssystem für homogene Polyprozessorsysteme
(Juni 1981)

Nr. 4 Akyildiz, J.F., Bolch, G.:
Analytic Solution Techniques for Queueing Network Models of Computer Systems
(Juli 1981)

Nr. 5 Lindstedt, W.G.:
Ein Verfahren zur Erstellung portabler algorithmischer Laufzeitprogramme
(Juli 1981)

Nr. 6 Vollmar, R.:
Zum Begriff des Parallelismus bei Polyautomaten
(Juli 1981)

Nr. 7 Luft, A.L.:
Rationaler Sprachgebrauch und orthosprachliche Standardisierung als Grundlagen zuverlässiger Software-Entwicklung und -Dokumentation
(Januar 1982)

Nr. 8 Händler, W.:
PROCEEDINGS OF A WORKSHOP ON TAXONOMY IN COMPUTER ARCHITECTURE
(Februar 1982)

Nr. 9 Beth, Th., Hess, P., Wirl, K.:
Materialien zur Kryptographie
(März 1982)

Nr. 10 Jahresbericht 1981 der Informatik
(März 1982)

Band 15
======

Nr. 1 Wittmann, A.:
Ein Mechanismus für die Synchronisation paralleler Prozesse
(Februar 1982)

Nr. 2 Maehle, E.:
Fehlertolerantes Verhalten in Multiprozessoren - Untersuchungen zur Diagnose und Rekonfiguration
(März 1982)

Nr. 3 Bathelt, P.:
Vergleich von Synchronisationsmechanismen
(Mai 1982)

Nr. 4 Keramidis, S.:
Eine Methode zur Spezifikation und korrekten Implementierung von asynchronen Systemen
(Juni 1982)

Nr. 5 Fromm, H.J.:
Multiprozessor-Rechenanlagen:
Programmstrukturen, Maschinenstrukturen und Zuordnungsprobleme
(Mai 1982)

Nr. 6 Bley, H.:
Vorverarbeitung und Segmentierung von Stromlaufplänen unter Verwendung von Bildgraphen
(Juli 1982)

Nr. 7 Hohl, W.:
Informatik-Sammlung
Katalog mit historischen Erläuterungen
(August 1982)

Nr. 8 Messerer, M.:
Ein neuer Ansatz zur Parallelisierung von Compilern
(August 1982)

Nr. 9 Grosch, J.:
Eine Programmiersprache mit mengentheoretischen Konstrukten und deren effiziente Implementierung
(August 1982)

Nr. 10 Kleinöder, W.:
Stochastische Bewertung von Aufgabenstrukturen für hierarchische Mehrrechnersysteme
(August 1982)

Band 15 Fortsetzung
==========

Nr. 11 Kneißl, F.:
Realisierung von Makro-Datenflußmechanismen auf hierarchischen Mehrrechnersystemen
(August 1982)

Nr. 12 Brendel, W., Fan, Z., Klar, R., Schmielau, W.:
ERES 82
Handbuch und Fallstudie
Handbook and Case Study
(Oktober 1982)

Nr. 13 Pelz, K.:
Portable Codegenerierung in Übersetzern für Prozeß-Programmiersprachen
(Oktober 1982)

Nr. 14 Sommer, U., Steinchen, J.:
Ein Kleinrechner-Grafiksystem zur Dokumentation von EDV-Abläufen
(Oktober 1982)

Nr. 15 Hein, H.-W.:
Das Erlanger Spracherkennungssystem
(November 1982)

Nr. 16 Bolch, G., Bruchner, W.:
Analytische Modelle für symmetrische Mehrprozessoranlagen mit dynamischen Prioritäten
(Dezember 1982)

Nr. 17 Jahresbericht 1982 der Informatik
(März 1983)

Band 16

Nr. 1 Gall, R.:
Formale Beschreibung des inkrementellen Programmierens-im-Großen mit Graph-Grammatiken
(März 1983)

Nr. 2 König, R.:
Beiträge zur algebraischen Theorie der formalen Sprachen
(März 1983)

Nr. 3 Welbourn, D.B.:
The Design of Mechanical Components and the Development of DUCT:
17 Years of CAD/CAM in Cambridge University
(April 1983)

Nr. 4 Woitok, R.:
Die System-Implementierungs-Sprache ALICE
(Mai 1983)

Nr. 5 Schneider, S., Schwab, P., Renninger, W.:
Wesen, Vergleich und Stand von Software zur Produktion von Systemen der computergestützten Unternehmensplanung
(Juli 1983)

Nr. 6 Händler, W., Herzog, U., Hofmann, F., Schneider, H.J. und Mitarbeiter:
Projektabschlußbericht 1978 - 1982 Erlangen General Purpose Array
(Juli 1983)

Nr. 7 Mackert, L.:
Modellierung, Spezifikation und korrekte Realisierung von asynchronen Systemen
(Juli 1983)

Nr. 8 Reitenspieß, M.:
Sprachkonstrukte zur Spezifikation und korrekten Implementierung von Schutzproblemen
(Juli 1983)

Nr. 9 Weber, K.:
Modellierung von Fehlverhalten mit Berücksichtigung paralleler Abläufe
(August 1983)

Nr. 10 Dietsch, H.:
Ein verteiltes Kommunikationssystem für einen Verbund aus lokalen Mikrorechner-Netzwerken
(Dezember 1983)

Band 16 Fortsetzung

Nr. 11 Maehle, E., Schmitter, E.:
Workshop Fehlertolerante Mehrprozessor- und Mehrrechnersysteme
(Dezember 1983)

Nr. 12 Volkert, J.:
Multiplikative Abstandsfunktionen und deren Anwendung in der Bildverarbeitung
(Dezember 1983)

Nr. 13 Steinbauer, D.:
Transaktionen als Grundlage zur Strukturierung und Integritätssicherung in Datenbank-Anwendungssystemen
(Dezember 1983)

Nr. 14 Wilke, P.:
Zusammenhänge und Unterschiede zwischen Graph-Grammatiken und Petri-Netzen sowie verwandter Systeme
(Dezember 1983)

Nr. 15 Jahresbericht der Informatik 1983
(März 1984)

Band 17

Nr. 1 Eberlein, W.:

Architektur technischer Datenbanken für integrierte Ingenieursysteme
(Februar 1984)

Nr. 2 Mertens, P., Heigl, M.:

Neuere Entwicklungen der computergestützten Produktionsplanung, Eignung - Verbindungen - Entwicklungspfade
(April 1984)

Nr. 3 Stoyan, H.:

Maschinen- unabhängige Code-Erzeugung als semantikerhaltende beweisbare Programmtransformation
(Juli 1984)

Nr. 4 Brendel, W.:

Funktionaler Entwurf und automatische Synthese hochintegrierter Schaltkreise
(August 1984)

Nr. 5 Brietzmann, A.:

Semantische und pragmatische Analyse im Erlanger Spracherkennungsprojekt
(Oktober 1984)

Nr. 6 Hoppe, B.:

Deterministisches Zuordnen vielfach durchlaufener Aufgabensysteme in Multiprozessorsystemen
(Oktober 1984)

Nr. 7 Fumy, W., Posch, S., Rieß, H.P.:

Materialien zur Codierungstheorie III
(Dezember 1984)

Nr. 8 Akyildiz, J. F.:

Leistungsanalyse von Multiprozessorsystemen mit Prozeßkommunikation
(Dezember 1984)

Nr. 9 Jahresbericht der Informatik 1984

(März 1985)

Band 18

Nr. 1 Siegmann, H.:

Wesen, Vergleich und Stand von zehn ausgewählten kleinrechnerorientierten Planungssprachen
(Februar 1985)

Nr. 2 Rathke, M.:

Parallelisieren ordnungserhaltender Programmsysteme für hierarchische Multiprozessorsysteme
(Juni 1985)

Nr. 3 Geus, L.:

Parallelisierung eines Mehrgitterverfahrens für die Navier-Stokes-Gleichungen auf EGPA-Systemen
(Juli 1985)

Nr. 4 Hellmold, K.-U.:

Der Simulationsrechner SIMPLEX
Eine Multiprozessorkonfiguration zur Simulation von zeitdiskreten Systemen auf GPSS-FORTRAN-Basis
(September 1985)

Nr. 5 Klebes, G.:

Entwicklung eines botschaftsorientierten Synchronisationsverfahrens für parallele Prozesse und seine Realisierung im Rahmen eines Realzeitprogrammiersystems
und
Rupprecht, G.:

Ein Konzept zur Integration von allgemeinen Sprachkonstrukten zur Synchronisation und von gemeinsamen abstrakten Datentypen mehrerer Prozesse in Modula-2 und seine Implementierung

Nr. 6 Schön, D.:

Technisches Informationssystem und Entwurf integrierter Ingenieurdatenbanken
(Oktober 1985)

Nr. 7 Vasconcelos, A.J.V. de:

Berechnung nichtstationärer Vorgänge in Drehstromnetzen mit Raumzeigerkomponenten auf einem Multiprozessor
(Dezember 1985)

Nr. 8 Baader, F.:

Die S-Varietät $\underline{DS}$ und einige Untervarietäten
(Dezember 1985)

Nr. 9 Jahresbericht der Informatik 1985
(März 1986)

Band 19
=======

Nr. 1 Erhard, W.:

Feldrechner DAP: Invertierung großer Matrizen
(Februar 1986)

Nr. 2 Fumy, W.:

Über orthogonale Transformationen und fehlerkorrigierende Codes
(März 1986)

Nr. 3 Ehrlich, U.:

Ein Lexikon für das natürlich-sprachliche Dialogsystem EVAR
(Mai 1986)

Nr. 4 Mühlfeld, R.:

Verifikation von Worthypothesen
(Juni 1986)

Offsetdruck · Fotodruck · Hogl · Hauptstraße 109 · 8520 Erlangen · Tel.09131/22990